Bachelor's Degree in Audiovisual System Engineering

BACHELOR THESIS

Design and development of mobile applications related with audio and radio (iOS platform)

Author : Álvaro Dueñas González

Tutors: Javier García Guzmán and Ignacio Gallego Pérez

Leganes, 15 de Julio de 2014

Index

Figure List

Table List

Chapter 1: Introduction

Chapter 1: Introduction

On this chapter I will introduce the problem to solve in this project, the motivation for solving it, the objectives that I have within the project and how I decided to do it. Additionally in this chapters there are two sections to provide information about some technical terms and about the entire memory.

1.1 Context

The number of mobile apps used have increased till 1.2 billions of people in 2012 , in 2017 the use rate will be 4.4 billions [1]. Under this situation of use of smartphones, communication has changed in a significant way.

Some new broadcasting ways, as podcasts and news feeds, compete with the traditional media, radio and newspaper. The competence has reached the point where traditional media is force to port their platforms to mobile environment.

The capability of smartphones to be connected all day and the large amount of provider bill rates, trigger an enormous change on communication.

The term social network is so popular and extended, becoming the way of identification to the rest of services. This allow environments not related with social networks that have adopted the advantages of them and that are complete integrated.

Under this social trend, every new app related with communication, have to absorb this new technologies and use them in their advantage and in user benefit.

In this scene, the development of platforms based on textual communication increased a lot, while audio communication have been relegated to a second level. Even in spite of the benefit of this kind of platforms, available every time and with high grade of listeners participation, there are no relevant improvements in the development of this platforms.

There are share audio content, but not under the same platform and in any case that allow comments of the same character, making them environments that not advance in a correct way.

1.2 Motivation of the project

The most important motivation of the project are the following:

- Develop a project related with audiovisual systems, according to the grade for witch I perform the project.
- Provide a platform that allows the distribution of information in audio, the iteration of the listeners to assess the quality of the published info and the possibility of comment in the same way as it was transmitted.
- From Journalist and Audiovisual Communication Department, the most important motivation is the implementation of new concept radio, based on the participation of listeners. The interaction of users is completely different with other platforms, the communication is more direct thanks to the spoken answers.

Other motivation is the large increase of mobile technologies in the last years. This technologies makes very easy and quick, the access to all kind of information. This fits perfectly with the main proposal of the project.

Finally the interest on develop my project with the research group SEL-UC3M, given that this group have developed some projects in the past for this platform and can provide me tools, advices and help, thanks to their experience.

According to this motivations this are the functions that have to be implemented in the project:

- A mobile application for iOS platform, providing the way to listen the updated audios and the way to create and upload new content for the platform.
- The platform have to provide an asynchronous model of communication, where the answer is not instantaneous, and that promote high quality content.
- A web service to manage all the audios uploaded to the platform, the user access and users permissions. As well as the organization of audios depending on the topic, with in a system of up-votes and down-votes provided by the users.
- A implementation of a radio for each of the topics of the platform, based on the most popular audios, in order to facilitate to the users a way to listen the best news of a specific topic.

1.3 Objetives

The main objectives of the project are the following:

- Develop an iOS app: All the services mentioned have to be used in a iOS application to allow the user interact with the web service, uploading audios, listening existing audios, voting according to their preference. This fits perfect with the goals from Journalist and Audiovisual Communication members of the project team. The idea of giving this tool to journalists, in order to establish a new kind of relation between radio and users, between journalists and listeners. Thanks to this application, users will interact with the radio, will create their own radio programs selecting audios in playlists and will participate in this new era of technologic radio.
- Implementation of a web service that manage the upload and download of the audios. This service has to provide a user identification to denote witch users can upload *main audios* (news) and witch ones only can upload *sub audios* (comments to the news). The web service will differentiate between the two main kind of users, content creators o journalist that can upload main audios and listeners that can comment the audios, uploading the comment with their own voice.
- Implementation of a system of votes for the audios in order to use this votes to create playlist of the best audios of each topic (Radios). Votes system not only is use to create the radios, in addition, it gives an important feedback to content creator, that allow them improve their work. Vote system helps a lot publishers work, that in the future will manage all audios to create radio streaming and that will filter toxic comments.

1.4 Method of resolution

The method followed to solve the has been iterative, because the not al the requirements were defined at the beginning and some of them arise from them evolution of the development. This implied that some new requirements emerged during the process and some other requirements were replaced or removed. The project evolved so much during the development looking forward the best approach to the objectives and to improve the user experience.

Most of the changes of requirements and goals, took place during meetings with the rest of the team. This meetings were defined to be every two weeks, and consist on the presentation of new implemented features and de discussion about some change in the implemented features or new features to be implemented. Meetings basically were based on the objectives of Journalist and Audiovisual Communication members, that try to explain the actual situation of the radio and how they want the app, because there is no other app that currently fulfill their goals.

At the end of the meetings some task were designed to the developer, according to the conclusions extracted from the discussions. Some task were planned to next meeting and the most complex ones had no fixed date to be perform and their evolution were followed during the meetings.

1.5 Terminology

This section includes a glossary that explain some terms and some abbreviations used along the document, in order to simplify reader understanding whole document.

1.5.1 Terms glossary

App: Way of denominate information applications designed form mobile devices,for example smartphones and tablets.

Android: Mobile operative system developed by Google. Android actually is the most used mobile operative system.

Django: Django is a free and open source web application framework, written in Python, which follows model-view-controller architecture pattern.

Framework: Set of classes that implement some functionalities and can be added to and application in order to this functionalities in an easier way.

iOS: Mobile operative system developed by Apple for their mobile devices (iPhone, iPad, iPod) and for AppleTV.

1.5.2 Abbreviations

AAC: Advanced Audio Coding is a standardized, lossy compression and encoding scheme for digital audio.

IDE: Integrated Development Environment, software application that provides facilities to computer programmers for software development of a certain platform. In this project the IDE used is Xcode which is the IDE of Apple.

JSON: JavaScript Object Notation, is an open standard format that uses text to transmit data objects consisting of key value couples. It is used to transmit data between a server and web application, as an alternative to XML. In this project JSON was chosen to communication between web server and the application [2].

MP3: MPEG-1 or MPEG-2 Audio Layer III, more commonly referred to as MP3, is an encoding format for digital audio which uses a form of lossy data compression [3].

MP4: Digital multimedia format used for encoding video and audio.

MVC: Model-View-Controller, MVC is the architecture pattern for implement user interfaces followed in most iOS applications. It divides a given software in three parts. The model is the central part of the app and contains logic, application data and functions. The view is the graphical part of the application. An the controller make possible the communication between model and view.

1.6 Content of the memory

In this section, there is a brief description of each chapter of this document.

In Chapter 1, Introduction, is explained the problem to be solved in this TFG, the motivation to develop the solution for the problem, the method followed in the project, a little explanation of some terms used along the document and a brief description of the memory contents.

In Chapter 2, State of the Art, is performed an a analysis of similar web and mobile applications with similar objectives than the ones explained in Chapter 1, and analysis of the iOS platform where the app is going to be developed, including the analysis of mobile technologies and devices.

In Chapter 3, Analysis of the application, is performed a description of the application to allow later in the chapter the explanation of user experiences and the indication of functional and non-functional requirements of the app. The description is a general view of the application, including the functions and a classification of type of users.

In Chapter 4, Application design, are defined the first and the last versions of paper prototypes used to model the final design of the application. Then shows the design alternatives, including the application and web services used, and components and classes diagrams. This chapter include the description of the data base used in the server as well as the description of the communication between the app and the server to interchange audios data.

In Chapter 5, Application testing, are defined some test done to the application in order to realize that cover of the functionalities expected.

In Chapter 6, Conclusion and future improvements, there is a conclusion of the author about the project and some author experiences. In addition it include a list of future improvements of the application, new features and improvements of actual functionalities.

Chapter 2: State of the art

Chapter 2: State of the art

Previous to the development of a project is needed to study the actual state of the problem to be resolved, and the possible implement solutions and existing applications, if any, that offer a solution to the problem or a similar one. Thereby, you obtain a vision of the whole problem and will know if a new contribution is needed.

The research of the existing applications, is motivated for the purpose of having a starting point instead of creating it from zero. Taking into account the good things detected to solve the problem and searching a solution for the detected deficiencies. The solution will suit the problem to be resolved.

In this chapter is exposed the information obtained from the research about the topics that will be attend during the development of the project. For this, firstly is exposed the analysis of the mobile and web applications related with the stated objectives. Then is exposed the development environment and its most important features. Finally, is exposed the way how the project have been developed.

2.1 Analysis of applications with same objectives

In order to approach to how some existing apps works in radio and audio threads, some existing apps were studied during team meetings. This analysis look for features that fits well in the objectives and features that will fill perfect but no one of the existing apps implement.

The analysis was performed to the following apps because audiovisual communication experts from team know them and have been using some of them from some time ago, so they know very well the strengths and weaknesses of each of them.

2.1.1 CuaQea

CuaQea is a new social network based on short audios that users upload to the platform speaking about their feelings, achievements, and stories [4].

CuaQea allow some little editions on the audios before upload them, like some filters. This makes more fun the app and approaches a to young audience. In addition allows interactivity with other social networks like Twitter and Facebook, with the possibility of sharing the audios in this other platforms.

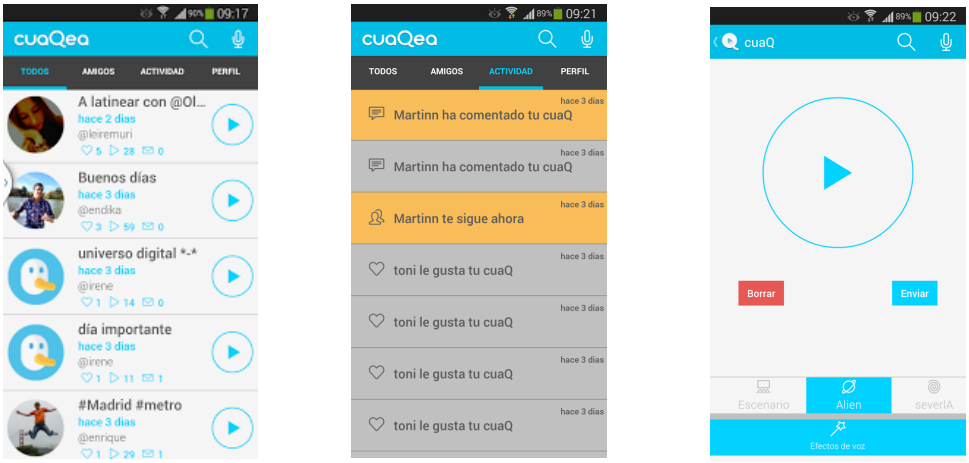


Figure 2.1 cuaQea

Important features of cuaQea:

- Audio upload
- Audio comments to main audios
- Share audios with social networks

2.1.2 Blaving

Blaving is social network that allows users share with everybody audios of 2 min of duration [5]. Blaving define itself as a new spoken Twitter. As in Twitter, users have a list of followers and a list of followed. Users can send direct messages and answer to other users post. Blaving interact with Twitter and allow sharing audios in this platform.

In addition, Blaving is implemented for iPhone, iPad, Android, Windows Phone, Blackberry and Java.

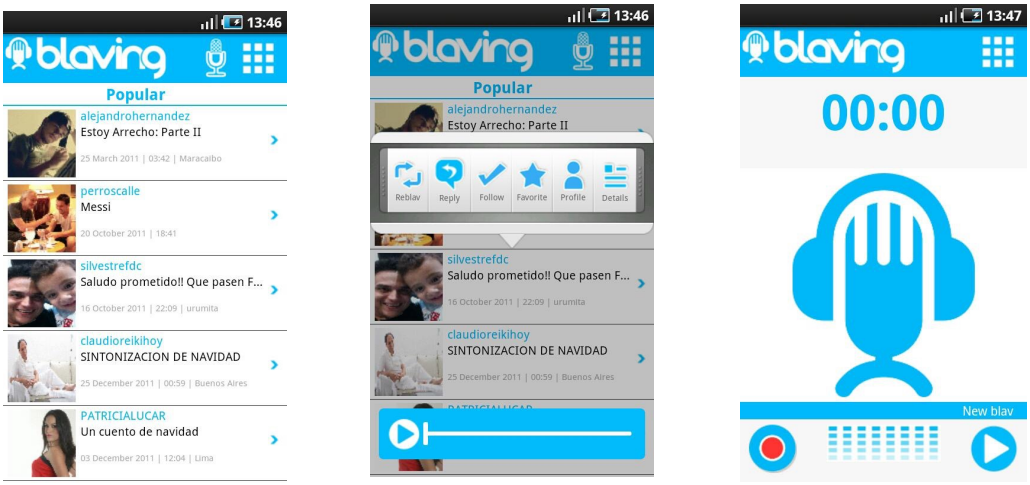


Figure 2.2 Blaving

Important features of Blaving:

- Audio upload
- Audio comments to main audios
- Share audios with social networks

2.1.3 SoundCloud

SoundCloud is a well known music web platform. SoundCloud has mobile apps for the most important mobile platform and is used by a lot of user to find new music and some artist to release new song [6].

SoundCloud allow users search music by author, by name… and follow music channels to discover new songs, create our own playlists or follow playlists of other users. As well allows user record new audios and share them in Twitter, Facebook and Tumblr.

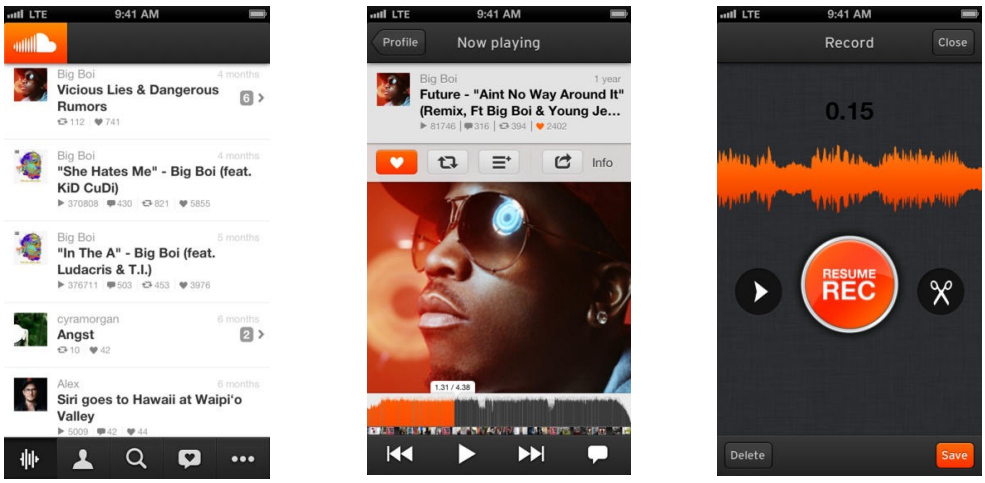


Figure 2.3 SoundCloud

Important features of SoundCloud:

- Audio upload
- Audios vote
- Creation of playlists
- Share audios with social networks

2.1.4 TuneIn

TuneIn is one of the most used apps to listen radios. It allows listen radios from whole world [7]. TuneIn has over 100,000 real radio stations and more than four million podcasts from all over the world.

Figure 2.4 TuneIn

Important features of TuneIn:

- Share audios with social networks

2.1.5 Audiovoo

Audiovoo allows recordings of 10 min for free that can be extended paying, and share this audios with everyone [8]. It have a web platform and allow users share the audios with Twitter and Facebook. Audiovoo have contents from BBC, Fox, The Guardian and other important partners.

Audiovoo allows the download of clips and playlist to reproduce them offline.

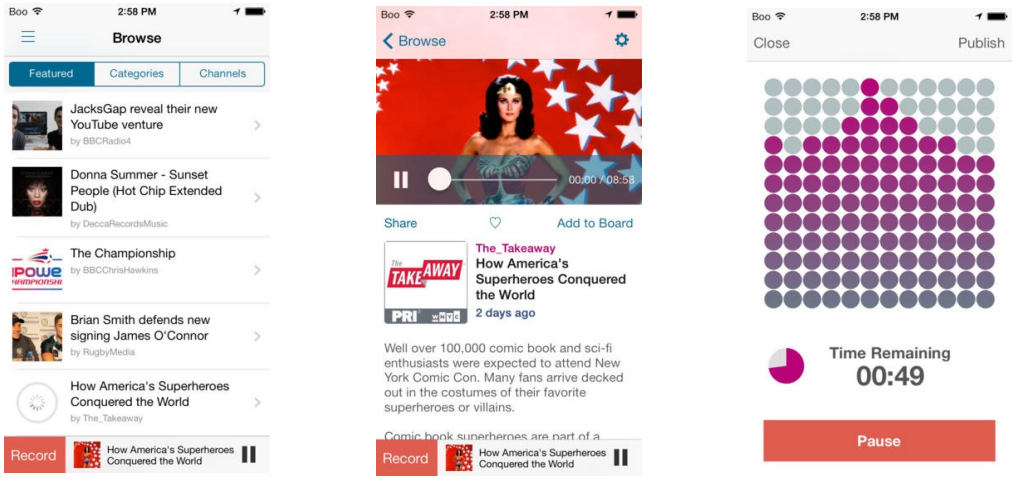


Figure 2.5 Audiovoo

Important features of Audiovoo:

- Audio upload
- Creation of playlists
- Share audios with social networks

2.1.6 Cadena SER for iPhone

This app allow listen to Cadena SER radio station programs with no interruptions [9]. Contains all the local city stations.Offer breaking news, videos, alternative live broadcasts, past recordings of your favorite programs and allow user share contents with social networks.

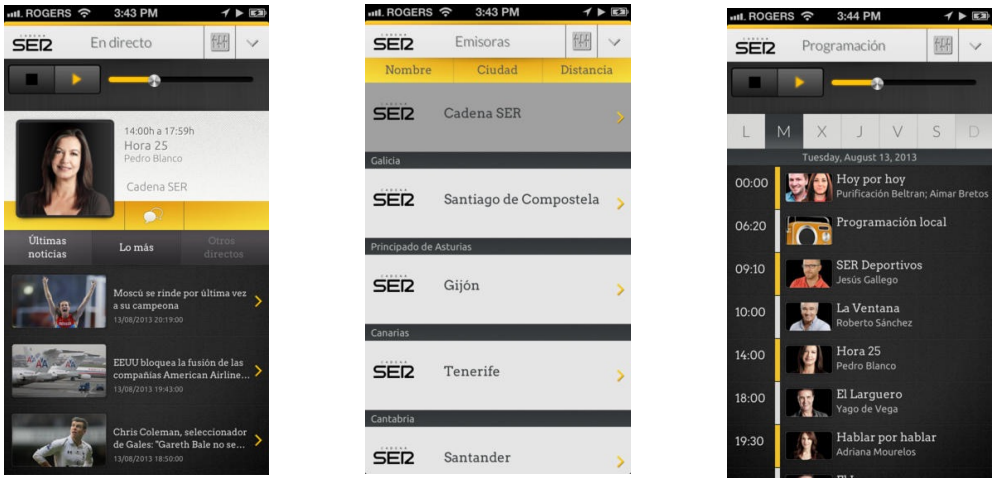


Figure 2.6 Cadena SER for iPhone

Important features of Cadena SER for iPhone:

- Audio vote
- Share audios with social networks

2.1.7 COPE Radio

COPE Radio allow listen to COPE in live [10]. Contains audios from all programs and allow users listen to them. In addition have information about local city stations and breaking news, videos.

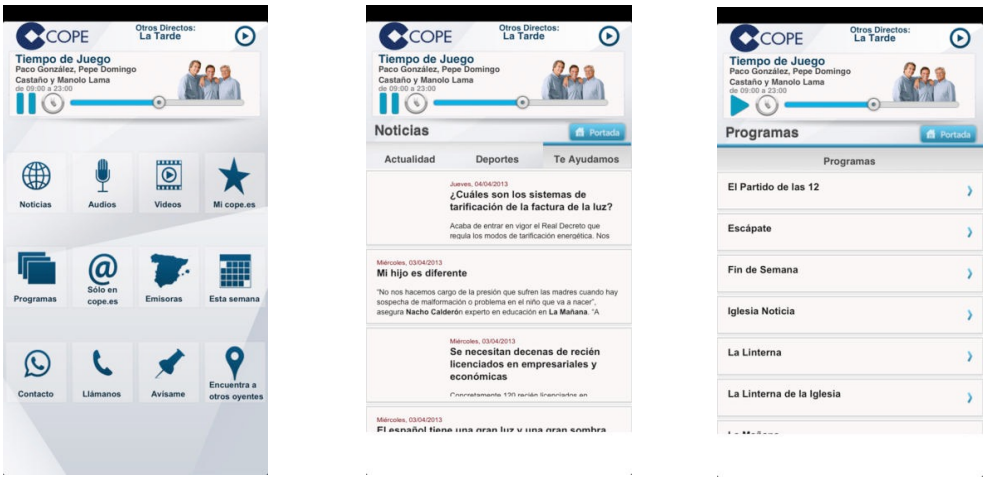


Figure 2.7 COPE Radio

Important features of COPE Radio:

- Audio vote
- Share audios with social networks

After explaining the characteristics of the analyzed apps, there is comparison table with the features that the project app will implement, in order to compare the apps between them and with the developed app Free Voices.

	cuaQea	Blaving	SoundCloud	TuneIn	Audiovoo	Cadena SER	COPE Radio
Upload audios	X	X	X		X		
Comment main audios via audio comments	X	X					
Vote audios			X			X	X
Create playlists			X		X		
Share with social networks	X	X	X	X	X	X	X

Table 2.1 App Comparison

2.2 Mobile development environment

Firstly in this section it will be analyzed the situation of the different mobile development environments. A general vision of iOS, the evolution and the internal structure. Finally is exposed the integrate development environment selected and the reasons.

As we can see in the article published by mobiThinking [1]: ”1.2 billion people worldwide were using mobile apps at the end of 2012. This is forecast to grow at a 29.8 percent each year, to reach 4.4 billion users by the end of 2017.”

Users of mobile apps worldwide by region 2012-2017 according to Portio Research			
	2012	**2013**	**2017**
App users worldwide	1.2 billion	N/A	4.4 billion
Asia Pacific	30%	32%	47%
Europe	29%	28%	21%
North America	18%	17%	10%
Middle East & Africa	14%	13%	12%
Latin America	9%	10%	10%
Source: © Portio Research (March 2013)		**via: © mobiThinking**	

Table 2.2 Users of mobile apps

According to the same article of mobiThinking, 56 billion smartphone apps will be downloaded in 2013. By operating system: 58 percent will be Google Android; 33 percent Apple iOS; 4 percent Microsoft Windows Phone; 3 percent BlackBerry.

As we can see in the graph of globalStats about internet traffic of mobile devices, iOS share with Android the first place .

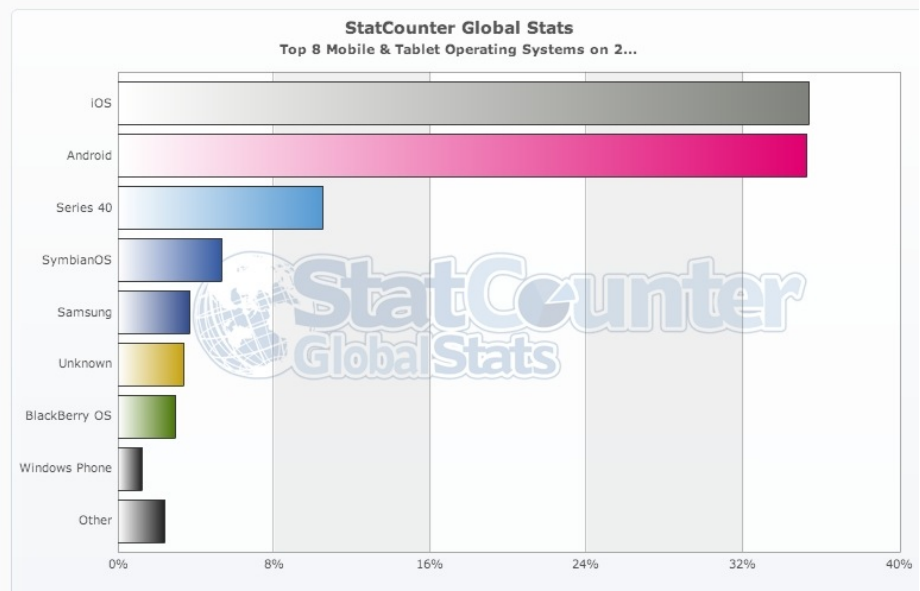


Figure 2.8 Internet traffic stats

2.2.1 General vision of iOS

iOs was created en 2007 with the launch of the first iPhone of Apple. The first version of the iPhone incorporated ,as operative system, iOS version 1.0. Apple have been incorporating iOS in all of their mobile devices, for example to the iPod Touch in 2007 with iOS 1.1 and the iPad in 2010 with iOS3.2.

IOS versions are numbered traditionally, the number in the left is the number of the version, next ones, separated by dots, correspond with the revision [11]. Each of the version doesn't have an specific name, as in Android, but they are know as iOS *X*, being *X* the number of the version.

Version	Devices	Launch date
1.0	iPhone	29/06/2007
1.1	iPhone iPod Touch	14/09/2007
2.0	iPhone 3G iPod Touch	11/07/2008
2.1	iPod Touch second generation	09/09/2008
3.0	iPhone 3GS iPod Touch second generation	17/06/2009
3.1.3	Last update for iPhone and iPod Touch of first generation	02/02/2010
3.2	iPad	02/04/2010
4.0	iPhone 4 iPhone 3GS and iPod Touch second generation (Limited improvements)	21/06/2010
4.2.1	iPhone 3G, iPhone 3GS, iPhone 4 iPod Touch 3er generation iPad	22/10/2010
4.3	iPhone 3GS, iPhone 4 iPod Touch 3th generation iPad and iPad 2	09/03/2011
5.0	iPhone 3GS, iPhone 4, iPhone 4S iPod Touch 3er generation iPad and iPad 2	12/10/2011
5.1	iPad 2 and iPad 3	07/05/2012
6.0	iPhone 5 iPod Touch 5th generation iPad 2, iPad 3, iPad 4 iPad mini	19/09/2012
7.0	iPhone 5S iPhone 5C iPad Air	10/06/2013

Table 2.3 iOS Evolution

In addition to use iOS for mobile devices, Apple included iOS as operative system for his second generation multimedia digital detector Apple TV. This are the most important updates of iOS for Apple TV.

Version	Launch date
4.1	04/09/2010
4.3	09/05/2011
5.0	12/10/2011
6.0	24/09/2012
7.0	28/01/2013

Table 2.4 iOS for Apple TV Evolution

As we can see in the tables, iOS is implemented in a large devices range, ranging from smartphones, touch music players and tablets, for which is possible to develop an app compatible with all of them.

Each of the updates has provided innovation improvements that take advantage of all the capabilities of the devices for which have been released, opening to the developers a very large amount of possibilities to include and combine, resulting in app with a lot of potential in the context for which they are designed.

2.2.2 iOS structure

In iPhone Application Development in 24 hours manual [12], is explained in detail, each of the layers of technology that compose the operative system iOS showed in the table, In this part , is explained each of them and and it function in the operative system.

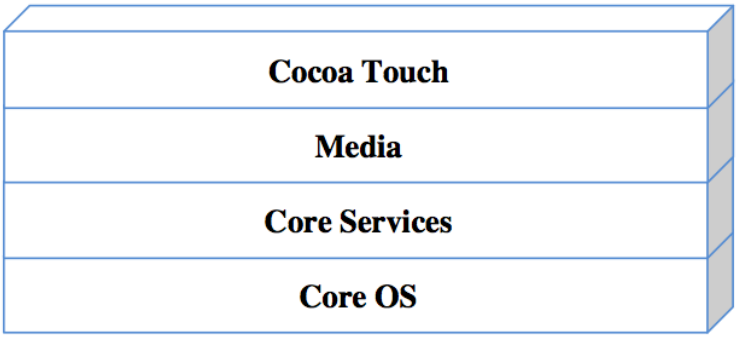


Figure 2.9 iOS Structure

Cocoa Touch

This layer is the highest level and allows the interaction of the user with the device, handling the elements that trigger income and outcome of data from and to the user. It can be said that provide the core of capabilities to the developed apps. In this layer is very important the framework UIKit because it includes resources for more that interface management .

Media

This layer is composed by frameworks that provide the creation of complex graphs, audio and video reproduction, and generation of 3D graphs in real-time.

Core Services

This layer is composed by frameworks that provide access to low level serves of the operative system. For example the access to files, network connection and types of objects. In this layer is very important the framework Foundation because provide most of the resources used of this layer.

Core OS

This layer is compose by frameworks that provide access to lowest level services of the operative system , for example handle threats, complex maths, external accessories and cryptography. The use of this functionalities is not necessary to develop general apps.

Frameworks used by most of the apps are UIKit from Cocoa Touch layer, CoreGraphics from Media layer and Foundation from Core Services layer, because with them most of the capabilities used by a developer are covered.

To develop iOS, Apple started for the Cocoa Touch layer from Cocoa, this grand a big mature to Cocoa Touch. Cocoa was born in middle of the decade of 80s in platform NeXSTEp and in 1996, Apple bought NeXT Computer. In the following years, Cocoa became to be the standard to develop applications for Macintosh.

The advantage to adapt Cocoa for iOS is double: the mature that iOS has versus other mobile operative systems and the easiness for Macintosh developers to start the development of iOS apps because of the similarity and utilization of the same patterns to develop in both platforms.

2.2.3 iOS apps life cycle

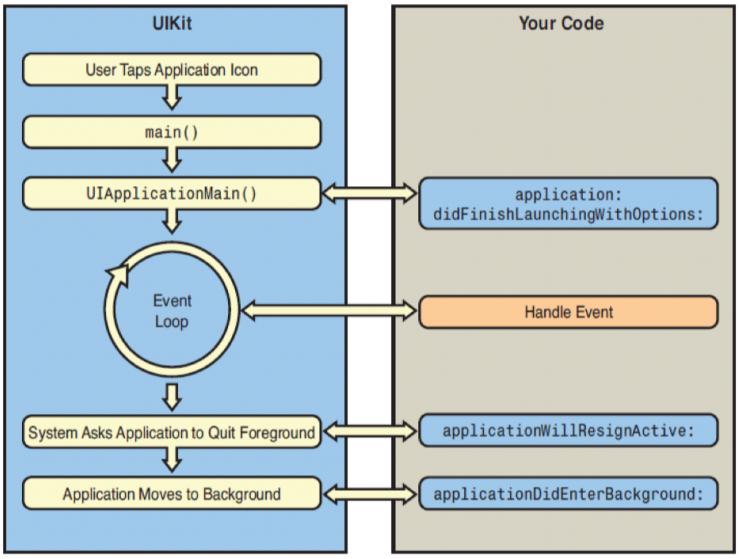


Figure 2.10 iOS apps life cycle

As we saw in iOS structure, framework UIKit provide resources to manage user interface and the events, as well as the management of execution of the app [13]. This way, when the suer init the app, the UIKit manage the initialization trough functions *main()* and *UIApplicationMain()*. If it is necessary, the developer have indicated in he code, the actions to perform when the app is initialized. This actions are defined in the method application:didFinishLaunchingWithOptions.

Once the app have been initialized, begins a boucle that receive all the event generated in the app. An example of and event is the tap of a interface button by the user, as every event, it is managed by the developer in the suitable methods.

The execution boucle continues until the request to the app to go background. Usually, it happens when the user press Home button of the device. When an app go to background, the execution does not finish and when the user select it at the init screen or at task manager, it come back to foreground mode at the same point of execution where it was before going to background. The method *applicationDidEnterBackground* make possible save the information needed to continue the execution of an app after coming from background mode and is different of the method *applicationWillResignActive* which is used in apps that can't go to background mode.

To sum up, the execution cycle of and app starts when the user select it at init screen. By default, there are functions that initialize the app and a method where the developer decide which actions perform in the initialization. When is initialized, the app go into a boucle that finish when the app go to background mode. Finally, while is in the boucle, it receive events which are managed by some methods wrote by the developer.

2.2.4 Integrated Development Environment

The IDE (Integrated Development Environment) used to develop the project is Xcode because is the only tool that allow create applications for Mac OS and iOS.

Xcode has been evolving since its launch and now is a very powerful development environment. It include the frameworks Cocoa and cocoa Touch, Xcode IDE and analysis tools. Only can be installed in computer with Mac OS. This are the principal parts of Xcode [14]:

Xcode IDE

Allows the build of the source code of the apps, the design of the user interfaces with the establishment of its relations with the code and debug the code, every thing in the same window. Gives help to edit the code, execute the apps to test them in a simulator or in a device and manage the licenses to sign the apps, that allows testing the apps and upload the apps to the App Store.

Apple LLVM Compiler

This compiler use the same parser to compile C/C++ and Objetive-C which is the language used to develop iOS apps. The compiler LLVM is integrated in the environment to identify what the developer is writing and offer ending of the code and identify syntax errors.

Instruments

This tools is used to performance and behavior analysis. The use is oriented to improve user experience in the apps and collect data related with memory or CPU usage in real time. That data is presented to the developer in a graph and is related with the code, in order to identify possible improvements in the app.

iOS Simulator

Used to save time testing apps, simulates the behavior of the app as if it installed in an iOS device. Allow to select the version and the device. But it has some limitation, it cant simulate some physical elements, for example cant simulate the camera.

The development of the project started on Xcode 5.0 and iOS7.

2.2.5 Objetive-C

Objective-C [15] is the programming language used to develop apps for MAC OS and iOS from Apple.

Objective-C is a object-oriented and hight level programming language. It appeared in 80s with the standardization ANSI of language C, and it is based on Smalltalk, one of the first object-oriented languages.

There are two kind of files, interface ones with .h extension and implementation ones with .m extension. .h files are used to define the attributes and the methods, and .m files are used to implement the methods using the attributes.

Objective-C use a specific kind of file to build the graphical user interface. This file has extension .xib and it is know as NIB. To access to the user interface elements, there defined in the interface files some links corresponding with the user interface elements. On NIB file there are defined the links and in the implementation files the developer use the elements connected with the links.

The differentiating feature of Objective-C is that it is a descriptive programming language. The definition of the method is focus on explain with the name, what is going to do. Using this feature, the code becomes easy to understand.

2.3 Analysis conclusions

Taking into account everything explained in this chapter, the reasons of the decision of using iOS to develop the project are the following:

- There is no app currently that fits perfectly in the objectives of the project in order to solve the problem. So the app will take advantage of strengths of analyzed apps and will solve the weaknesses of them.
- iOS is the second most popular mobile platform according to the number of apps downloaded in last year. That implies that the platform provide the app a good way to get to the users.
- iOS share with Android the first place of internet mobile traffic, having much less terminals, that implies that iOS users use more their mobile devices.
- Additionally, this project is being developed in parallel for Android platform.

Chapter 3 : Requirements specification

Chapter 3 : Requirements specification

3.1 Introduction

This chapter includes a description of the system developed, user records and software requirements. For its realization have followed the parts stablished at standard 830 of IEEE.

The general description provide more specific information about the development system that provide a solution to the problem introduced in Chapter 1. Furthermore, helps to introduce user records and software requirements.

User records are defined to know the interactions of the users with the system. Each record contains a list of task that the user have to perform in order to obtain a functionality from the system, this is the reason to include in each of the records a indication of the functional requirements and not functional, related with the record.

Finally, after the analysis of the tasks to be executed by the system and how are going to be done, a collection of functional and non-functional requirements of the system is performed.

3.2 General description of the system

Following the standard IEEE 830, this section describe all the factors that affect to the system and the requirements. The goal of this section is describe the context of the requirements , done in section 3.4, and make easier to understand them.

3.2.1 Product perspective

The project arise of the idea of developing a platform intended to manage audios of short duration, in principal with informative and news content. The platform is oriented to some expert users that have the competence of creating content (main audios) and upload it to the platform, and to the rest of the users, that can access to all the content, can upload audios with their opinion about a concrete audio, but can’t upload main audios.

iOS app will coexist with an Android app that is being develop at the same time. This affect the project in the way that part of the server implemented for the application is already designed. In fact, Android developer choose Django as a framework to design the web service. In Chapter 4 there is a explanation about Django server implementation.

The platform is has two main purposes:

- Provide to web magazines, web radios, blogs, radios… In general to everybody dedicated to create informative content, a way to present their content, in an easy way to everybody.
- Provide to information consumers, an easy and compact way, to access to informative content of a lot of journalist, bloggers… And the access to this content classified in tags.

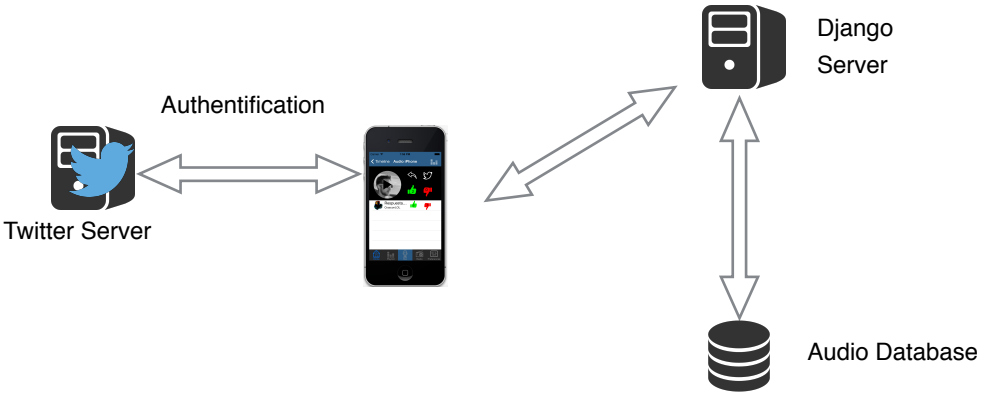


Figure 3.1 Product scheme

3.2.2 Product functions

Now that the problem ,the objectives and the solutions have been exposed, is the moment to expose the main functionalities of the developed app:

- **Timeline:** User will be able to see all the audios uploaded to the platform in chronological descendent order, in other words, last update will be the first audio that the users will see.
- **Reproduction**: Users will be able to download the audios and listen to them. As well as voting them positively or negatively. In addition users will be able to listen opinions of other users, the replies.
- **Reply**: Users will be able to reply to the audios creating their own audio, which can be seen and vote by all the users.
- **Upload audios**: “Premium” users will be able to create their content in the app and unload it to the platform. As well as classify that content depending on the content, selecting the tag that fits better with the audio.
- **Select Radios**: The app create one radio with the ten most voted audios of each tag, users will be able to set which radios have on their phone.

3.2.3 Interaction with other systems

The application interacts with other systems which are fundamental for its correct performance. The application should manage the situation of one or both of them down. The systems are the following:

- **Free Voice Server**: This server is provided by the university, and manage the audio and audio data storage.
- **Twitter Server**: The app use twitter servers to identify the user as well as obtain its user id and profile photo.

3.2.4 User characteristics

The application is oriented for three different groups of users:

- **Journalists**: This is group is composed by users that will upload the main content, this users can answer audios as well as upload new main audios. This group is defined for journalist that can upload their opinion about important events and news.
- **Common Users**: This group is composed by users that will listen to the audios and will comment them. Most of the users will be in this group.
- **Publishers**: This group will be composed by user that will create the radio streaming when it will be implemented. This users will generate the radios taking main audios and answers to those audios.

3.2.5 Restrictions

In order to establish the limits of the system and record some aspects of the system, the following restrictions have been determined:

The user interface of the app have to fulfill the design and development guides of Apple.

The application have to be compatible with devices with iOS version 7 or higher.

The programing language have to be Objective-C

The communication between Django Server and the app have to be trough a web service that return the data in JSON format and the app have to process that data.

The application have to implement a way of sharing the audios using Twitter.

3.2.6 Assumptions and dependencies

The application depends in some external services, the ones described on 3.2.3. This implies that the app depends directly from this services and create a very important dependency of them to operate correctly.

The principal dependencies of the app are the following:

- **Django Server**: Incase that this server is down, the app can not obtain the info of the audios, neither the audios.
- **Twitter Identification Service**: Incase that this service is down, the app can not identify the user so the user can not interact.

3.3 User stories

This section use user stories to detail, in a handy language to every people out of developing systems, the functionalities of the developed app should provide. In SCRUM [16] methodology, it is use to describe the characteristics that the user expect from the system.

User stories are sequences of the user interaction with the system. This interactions are decomposed in task that have to performed, in order to access and/or get the functionality that the user wants to use. User stories have to be independent, negotiable, estimable, small and ascertainable.

User stories can incorporate aspects related with system functionality, as well as any other aspect of the system, because they are performed form user point of view.

The table format used in this document, to describe user stories, is the following:

ID	**EFFORT**
OBJECTIVE	
STEPS SECUENCE	
FUNCTIONAL REQUIREMENTS	
NON FUNCTIONAL REQUIREMENTS	

Table 3.1 Table format for User Stories

Each field will have this content:

- ID: User story identifier. With format US-XXX
- EFFORT: Time expend, in hours, to implement the functionality.
- OBJECTIVE: Functionality of the system that satisfy.
- STEPS SEQUENCE: Steps that the user has to perform to reach the functionality.
- FUNCTIONAL REQUIREMENTS: Functional requirements extracted from the user story.
- NON FUNCTIONAL REQUIREMENTS: Non functional requirements extracted from the user story.

ID : US-001	**EFFORT**
OBJECTIVE	Listen to an uploaded audio.
STEPS SECUENCE	1-Init the app 2-Select Twitter account 3-Select timeline tab, in Timeline view, appear all audios identified by title, author and photo, this compose an audio cell 4-Select the audio cell that launch Reproduction view 5-Click play button in Reproduction view
FUNCTIONAL REQUIREMENTS	FR-001,FR-003,FR-004,FR-007
NON FUNCTIONAL REQUIREMENTS	IR-001,IR-006

Table 3.2 US-001 Listen uploaded audio

ID : US-002	EFFORT
OBJECTIVE	Listen to an uploaded audio answer of an specific audio.
STEPS SECUENCE	1-Init the app 2-Select Twitter account 3-Select timeline tab, in Timeline view, appear all audios identified by title, author and photo, this compose an audio cell 4-Select the audio cell that launch Reproduction view 5-Click answer audio cell to download the audio 6-Clic answer audio play button
FUNCTIONAL REQUIREMENTS	FR-001,FR-002,FR-003,FR-004,FR-007
NON FUNCTIONAL REQUIREMENTS	IR-001,IR-006

Table 3.3 US-002 Listen uploaded audio answer

ID : US-003	EFFORT
OBJECTIVE	Upload new audio.(Only authorized usres)
STEPS SECUENCE	1-Init the app 2-Select Twitter account 3-Select rec tab 4-Select audio tag from the picker with actual tags of the server 5-Click rec button to start the recording and stop rec button to finish the recording 6-Click upload button that launch an alert asking the user to set a title for the audio 7-Set title for the audio and click upload option
FUNCTIONAL REQUIREMENTS	FR-004,FR-006
NON FUNCTIONAL REQUIREMENTS	IR-003

Table 3.4 US-003 Upload new audio

ID : US-004	EFFORT
OBJECTIVE	Upload new audio answer
STEPS SECUENCE	1-Init the app 2-Select Twitter account 3-Select timeline tab, in Timeline view, appear all audios identified by title, author and photo, this compose an audio cell 4-Select the audio cell that launch Reproduction view 5-Click answer button that launch Recording view 6-Click rec button to start the recording and stop rec button to finish the recording 6-Click upload button that launch an alert asking the user to set a title for the audio 7-Set title for the audio and click upload option
FUNCTIONAL REQUIREMENTS	FR-001,FR-004,FR-006

ID : US-004	EFFORT
NON FUNCTIONAL REQUIREMENTS	IR-001,IR-006,IR-003

Table 3.5 US-004 Upload audio answer

ID : US-005	EFFORT
OBJECTIVE	Listen to a radio playlist
STEPS SECUENCE	1-Init the app 2-Select Twitter account 3-Select radio tab 4-Swipe right or left to find the desire radio. 5-Select the audio cell that launch Reproduction view 6-Click play button
FUNCTIONAL REQUIREMENTS	FR-003,FR-009
NON FUNCTIONAL REQUIREMENTS	IR-002,IR-006

Table 3.6 US-005 Listen to a radio

ID : US-006	EFFORT
OBJECTIVE	Set desire radios
STEPS SECUENCE	1-Init the app 2-Select Twitter account 3-Select preferences tab 4-Select and deselect radio switches
FUNCTIONAL REQUIREMENTS	FR-010
NON FUNCTIONAL REQUIREMENTS	IR-005

Table 3.7 US-006 Set radios

3.4 Specific requirements

The requirements defined in this section have the level of detail required for designers to implement a system that satisfy the requirements and allow the testing team plan and perform test that show if the system fulfill the requirements. The requirements define external behaviors of the system noticeable by the users, operators and other systems.

Section 3.4.1 define the functional requirements that explain the functionalities implemented by the app.

Section 3.4.2 define requirements of eternal interfaces.

Section 3.4.3 define requirements that affect to the performance of the app.

Section 3.4.4 define design restrictions that limit some aspects of the app.

To table used to define the requirements is the following:

ID		NAME	
DEPENDENCIES			
DESCRIPTION			

Table 3.8 Format table for Specific requirements

Each field will have this content:

- ID: User story identifier. With format: FR-XXX for functional requirements, IR-XXX for external interface requirements, PR-XXX for performance requirements, DR-XXX for design restrictions.
- NAME: Name of the requirement. The name summarize the requirement, the action, the complements, and who perform the action.
- DEPENDENCIES: identifier of the requirements needed to be implemented.
- DESCRIPTION: explanation of the requirements that allow any developer to implement it with out consulting any other information.

3.4.1 Functional requirements

ID	FR-001	NAME	User record new audio (Only authorized users)
DEPENDENCIES			
DESCRIPTION	The app allow authorized users to record an audio clip and upload it to the platform. Users give a tag and a title to the audio before uploading it and the app notify the user that the audio have been uploaded correctly.		

Table 3.9 FR-001 User record new audio

ID	FR-002	NAME	User record new audio answer
DEPENDENCIES	FR-001		
DESCRIPTION	The app allow users to record an audio clip to answer other audio and upload it to the platform. Users give a title to the answer before uploading it and the app notify the user when the audio is uploaded.		

Table 3.10 FR-002 User record new audio answer

ID	FR-003	NAME	User reproduce main audio/answer audio
DEPENDENCIES	FR-001,FR-002		
DESCRIPTION	The app allow users to reproduce any main audio and any answer audio. Audios are identify by title and author name, so the user can chose.		

Table 3.11 FR-003 User reproduce audio

ID	FR-004	NAME	User set audio name
DEPENDENCIES	FR-001,FR-002		
DESCRIPTION	The app allow users to give a short description of the audio to upload, and it is use as title for the audio.		

Table 3.12 FR-004 User set audio name

ID	FR-005	NAME	User votes audios
DEPENDENCIES	FR-003		
DESCRIPTION	The app allow users to vote positive or negative all audios.Each main audio or answer audio, have to buttons to vote,only one of them can be pushed at same time, so users can not vote positively and negatively the same audio.		

Table 3.13 FR-005 User votes audios

ID	FR-006	NAME	User set audio tag
DEPENDENCIES	FR-001,FR-002		
DESCRIPTION	The app allow users to set a tag to their audios that will be used to organize audios depending on the tags and create the radios. In Record view, there is a picker with the actual tags from the server, and users have to chose one.		

Table 3.14 FR-006 User set audio tag

ID	FR-007	NAME	Platform identify audio author
DEPENDENCIES	FR-001,FR-002,FR-003		
DESCRIPTION	The system identify the authors of the audios using Twitter accounts. The app connect to Twitter service and obtain users info based on its Twitter id.		

Table 3.15 FR-007 Platform identify audio author

ID	FR-008	NAME	App set twitter image to an audio
DEPENDENCIES	FR-001,FR-002,FR-003		
DESCRIPTION	The app identify the audios with the Twitter image of the author. The app connect to Twitter service and download the profile photo of the user with the Twitter id.		

Table 3.16 FR-008 App set twitter image to an audio

ID	FR-009	NAME	User creates a playlist
DEPENDENCIES	FR-001,FR-002		
DESCRIPTION	The app allow users to create a playlist to reproduce audios in the order they want. Users can select an audio for a play list pushing the button playlist in Reproduction view and listen to the playlist in Playlist view.		

Table 3.17 FR-009 User creates a playlist

ID	FR-010	NAME	User select radios
DEPENDENCIES	FR-006		
DESCRIPTION	The app allow users to selects which radios appear in the radio tab. In Preferences view there are some switches to select the radios.		

Table 3.18 FR-010 User select radios

3.4.2 External interfaces

ID	IR-001	NAME	Timeline view
DEPENDENCIES	FR-001		
DESCRIPTION	The app should provide a view where appear all the main audios ordered by upload date.		

Table 3.19 IR-001 Timeline view

ID	IR-002	NAME	Playlist view
DEPENDENCIES	FR-009		
DESCRIPTION	The app should provide a view with the audios added to the playlist by the user, ordered in the order specified by the user.		

Table 3.20 IR-002 Playlist view

ID	IR-003	NAME	Record view
DEPENDENCIES	FR-003		
DESCRIPTION	The app should provide a view where the user can record new audios and upload them. In addition it should have controls to reproduce the audio before the upload, controls to select the tag and a dialog to set the title before upload the audio.		

Table 3.21 IR-003 Record view

ID	IR-004	NAME	Radio view
DEPENDENCIES	FR-005,FR-006		
DESCRIPTION	The app should provide a view with the radios selected by the user and each radio contain the ten most voted audios of the radio tag.		

Table 3.22 IR-004 Radio view

ID	IR-005	NAME	Preferences view
DEPENDENCIES	IR-004,IR-006,FR-005,FR-006		
DESCRIPTION	The app should provide a view with controls to select the desired radios, as well as select if the answer audios appear ordered by votes or by update date.		

Table 3.23 IR-005 Preferences view

ID	IR-006	NAME	Reproduction view
DEPENDENCIES	IR-005		
DESCRIPTION	The app should provide a view with: -Reproduction, vote, answer and share controls for the selected audio -List of the answer audios ordered depending on the preferences option -Controls to download, reproduce and vote answer audios		

Table 3.24 IR-006 Reproduction view

3.4.3 Performance requirements

ID	PR-001	NAME	Fluid UI
DEPENDENCIES			
DESCRIPTION	Starting from that the app runs in an iPhone with normal performance and a stable internet connection, the app should provide fluid UI when the app is synchronizing with the server.		

Table 3.25 PR-001 Fluid UI

3.4.4 Design restrictions

ID	DR-001	NAME	Implementation compatible with some iOS versions.
DEPENDENCIES			
DESCRIPTION	The app should be compatible with Apple devices that use iOS 7 or greater.		

Table 3.26 DR-001 Compatible with iOS versions

ID	DR-002	NAME	Internet connection necessary.
DEPENDENCIES			
DESCRIPTION	The app need internet connection to identify the user before entering in the app. If there is no internet connection the app can not identify the user and the app can not be used.		

Table 3.27 DR-002 Internet connection

ID	DR-003	NAME	Audio codex compatible with iOS.
DEPENDENCIES			
DESCRIPTION	Audios uploaded have to use a codex compatible with iOS to be reproduced. https://developer.apple.com/library/ios/documentation/audiovideo/conceptual/multimediapg/usingaudio/usingaudio.html		

Table 3.28 DR-003 Audio codex

3.4.5 System atributes

The app have to be reliable and have to take into account all possible errors and show information related to the user, in other to solve them if depends on the user.

The app have to be easy to maintain by other developer using the available documentation.

The app have to be protected from attacks, in this case the security is implemented in the server, so in the app is not necessary to implement security aspects.

Chapter 4 : Application design

Chapter 4 : Application design

This chapter include the paper prototype designed at the beginning of the project. Paper prototypes are useful to express graphically the requirements of the app, this allow to show the aspect of the app and how the user will interact with the app.

The components diagram gives a whole view of the design and the classes diagram show the relations between the classes involved in components diagram.

Sequence diagrams are necessary to allow a developer, the implementation of the interaction of the system parts.

4.1 Mobile Application design

4.1.1 Interface design

First paper prototype design

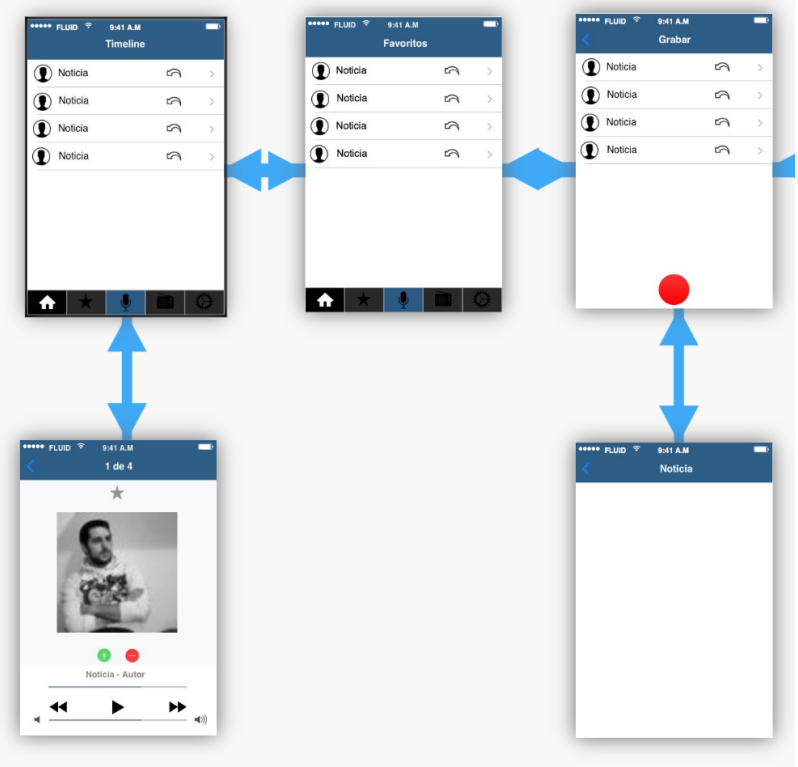


Figure 4.1 First paper prototype

The first version of the paper prototype shows the most important views of the application, Timeline, Favorites, Record and Reproduction.

Timeline: The user can see all the audios uploaded to the server and can answer them directly from this view.

Favorites: The user can store all the audios that he/she likes and listen to them later.

Record: The user can record a new audio and upload it, and can see the rest of the audios uploaded before.

Reproduction: This view contain the common media controls as well as vote controls. The audios is identity with author's twitter image. And with the favorite button, the audio can be added to favorites.

Final design

This version of the paper prototype is the final design of the app, it has some modifications from the first prototype.

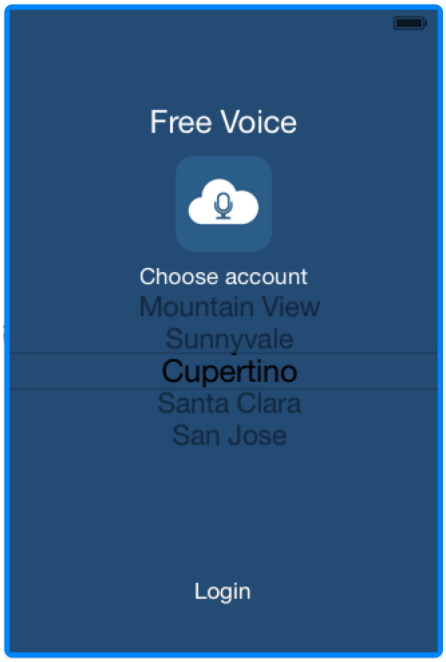


Figure 4.2 Login view

This view did not appear in the first paper prototype, is the initial view and where the user selects which twitter account from the iPhone will use to connect to the app.

Timeline view is similar to the first prototype, the only change is the disappearance of the answer button, that now is on Reproduction view.

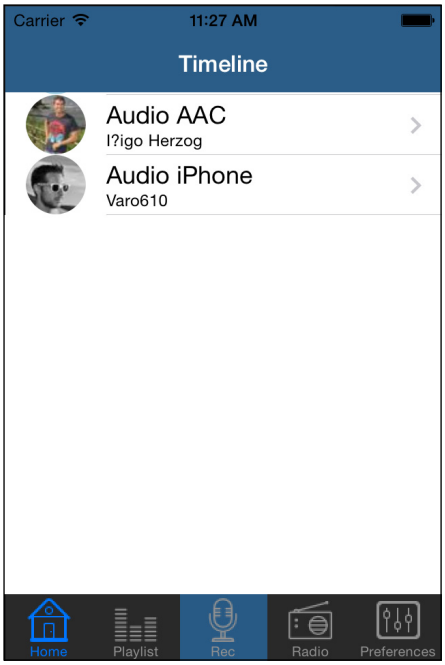


Figure 4.3 Timeline view

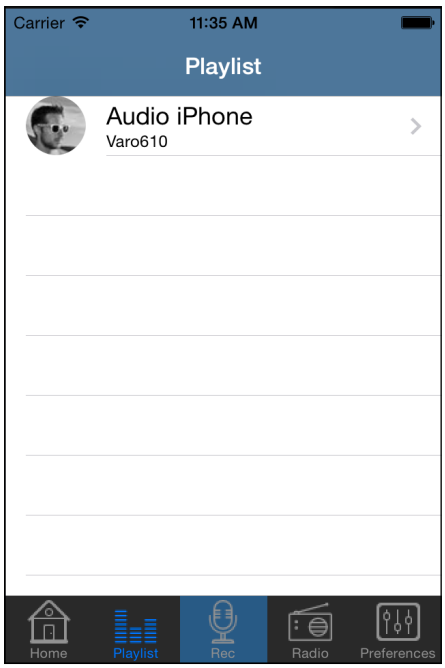


Figure 4.3 Playlist view

Favorites view from first prototype evolved to Playlist view, this change was induced by the type of content of the audios. As the content will be news, it makes little sense store past news, but now this view allow the user to create a playlist with the audios that is interested, while listening to other audios.

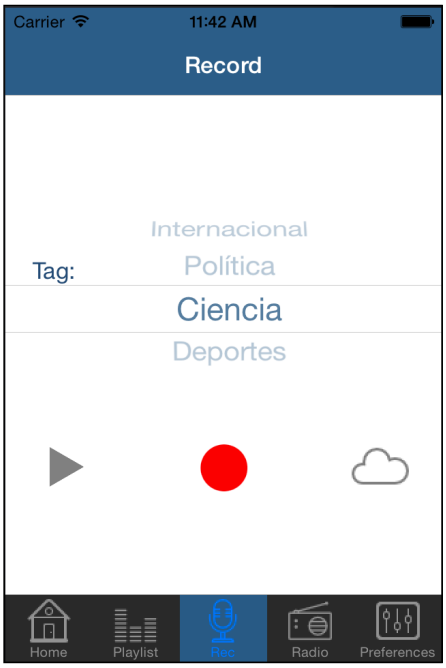


Figure 4.4 Record view

Record view has some changes from first prototype. Past uploaded audios do not appear any more in this view. The user select the tag related with the audio and has the controls for record, listen and upload the audio.

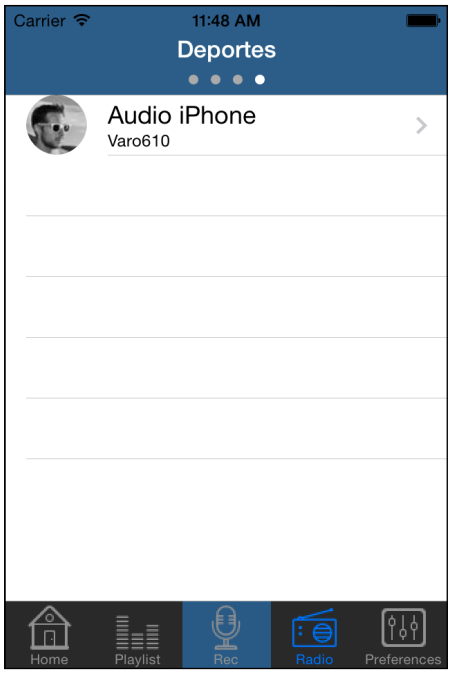


Figure 4.5 Radio view

Radio view did not appear in the first prototype, this view implement the service where the user find the ten most voted audios of each tag. The user can swipe left or right to select different radios.

Preferences view did not appear in the first design. This view provide to users the controls to select which radios appear in Radio view and the order of the answer audios in Reproduction view.

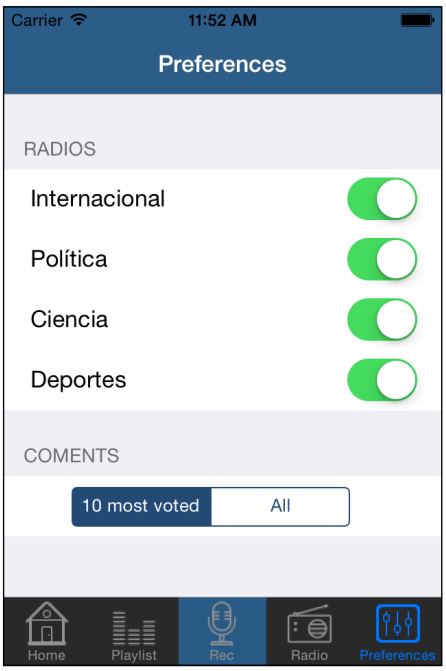


Figure 4.6 Preferences view

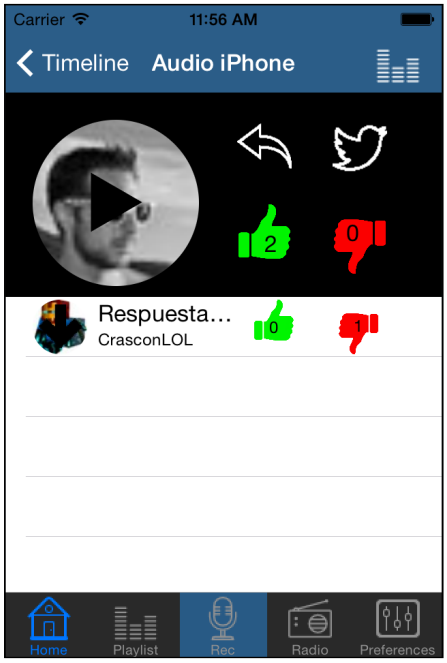


Figure 4.7 Reproduction view

Reproduction view differs a lot from first design. Media controls have been reduced to just one button Play/Stop. Now users can answer the audio after listening the audio and can share the audio using Twitter.

Following the preferences of the user, selected in Preferences view, in the bottom appear answer audios with in the controls to download, reproduce and vote them.

In the top right there is the button to add the audio to the playlist.

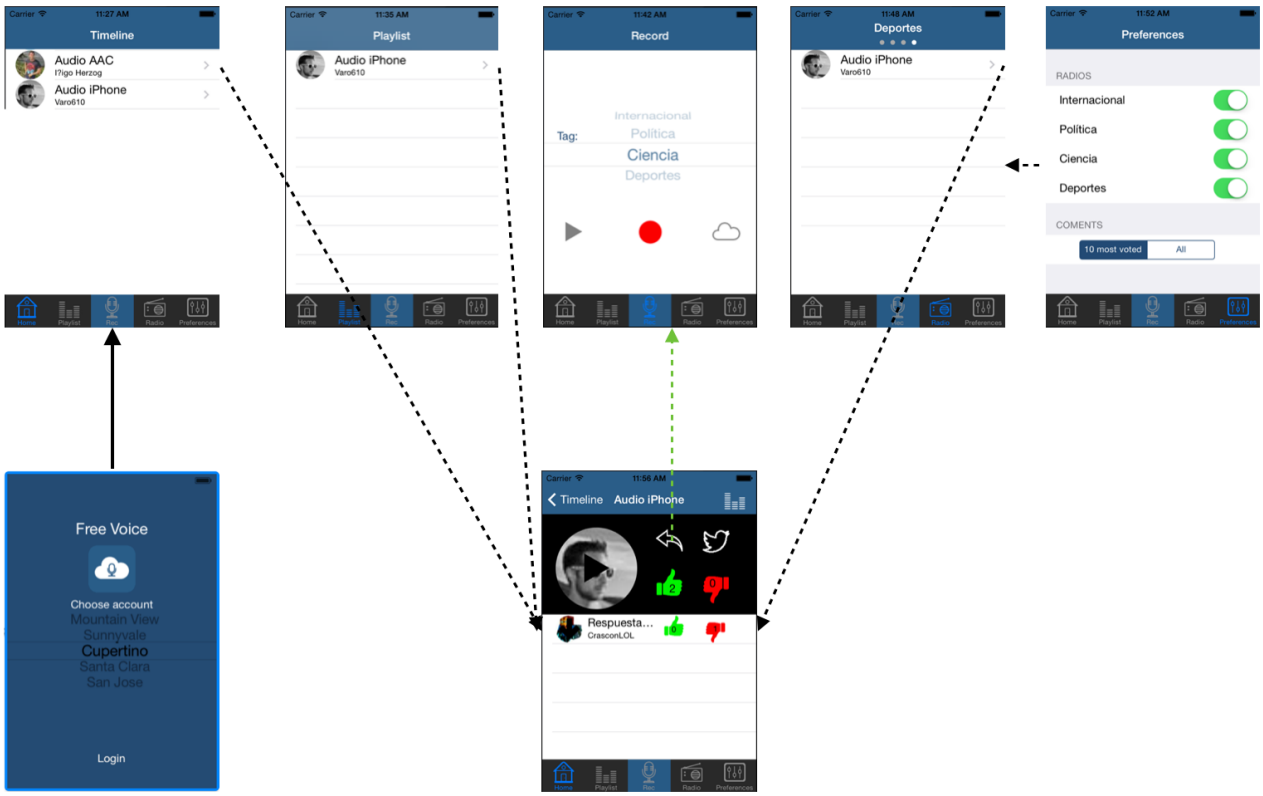


Figure 4.8 Final design

Main changes

During the evolution of the project, there where meetings with the rest of the team and some aspect change from the first design finally creating the final design. These are the main changes:

- Answer button was moved from audio cell in Timeline view to Reproduction view.
- Favorite tab evolve to Playlist tab in order to fit better in the proposals of the app.
- User past uploads do not appear any more in Rec view, this made the recording process more intuitive.
- Preferences radio section added in order to allow user select radios.
- Preferences comments section added to allow user select in which form appear the answers.
- Answers audios appear in Reproduction view and can be reproduced directly there, this makes better the user experience.

4.1.2 Design alternatives

This section shows the different alternatives taken into account during the design phase and the ones chosen in the final design.

Inside the different alternatives of design there are some system elements with different options of the design that establish the design alternatives.

System Elements

Web service: At the beginning of the project a part of the web service was already designed, because the application is being developed for Android platform platform, and the web service have to be common to allow the interaction of users from both mobile platforms. The web service is implemented using Django and the language chosen to retrieve the data from the server was JSON because JSON provide very fast responses and low amount of bytes to download.

Retrieved data format: Following Django standards the data is retrieved according to the models defined. In this case there are three models and format is the following:

Main Audios:

```
{
     id: 3,
     description: "Audio iPhone",
     user: "Varo610",
     userid: 136669759,
     timestamp: "05/27/2014 19:28:31",
     upvotes: 2,
     downvotes: 0,
     tags: [
          "http://163.117.154.146/bvs/tags/4/"
     ],
     url: "http://163.117.154.146/bvs/mains/3/",
     contents: "http://163.117.154.146/bvs/media/documents/
2014/05/26/Audio_iPhone.mp4"
}
```

Sub Audios:

```
{
     id: 2,
     parent: "http://163.117.154.146/bvs/mains/3/",
     parent_id: 3,
     description: "Respuesta iPhone",
     user: "Varo610",
     userid: 2192595405,
     timestamp: "06/05/2014 09:55:53",
     upvotes: 0,
     downvotes: 1,
     tags: [
          "http://163.117.154.146/bvs/tags/4/"
     ],
     url: "http://163.117.154.146/bvs/subs/2/",
     contents: "http://163.117.154.146/bvs/media/documents/
2014/05/26/Respuesta_iPhone.mp4"
}
```

```
Tags:
{
      id: 1,
      keyword: "Internacional",
      url: "http://163.117.154.146/bvs/tags/1/"
}
```

Audio format: In order to facilitate the compatibility of the audios with both platforms (iOS and Android), there were two main options. MP3 and AAC encapsulated in MP4, finally the decision was AAC encapsulated in MP4 taking into account that nowadays MP4 is the most extended audio and video format on Internet.

Author profile photo: Given that the application use Twitter to identify users, there exist the possibility of identify users in the app not only using its Twitter name, using their profile photo. Finally the design have the author profile photo, in order to make easier the identification of author graphically.

First design alternative

The first alternative consist on obtaining al information from the web service and implement a data base in the telephone to process the data. The data is retrieved in JSON from the server and transfer to the DB by the iPhone.

This alternative was discarded because it will work fine when the amount of data is not to big, but finally it will slow down the performance of the application and will deteriorate user experience.

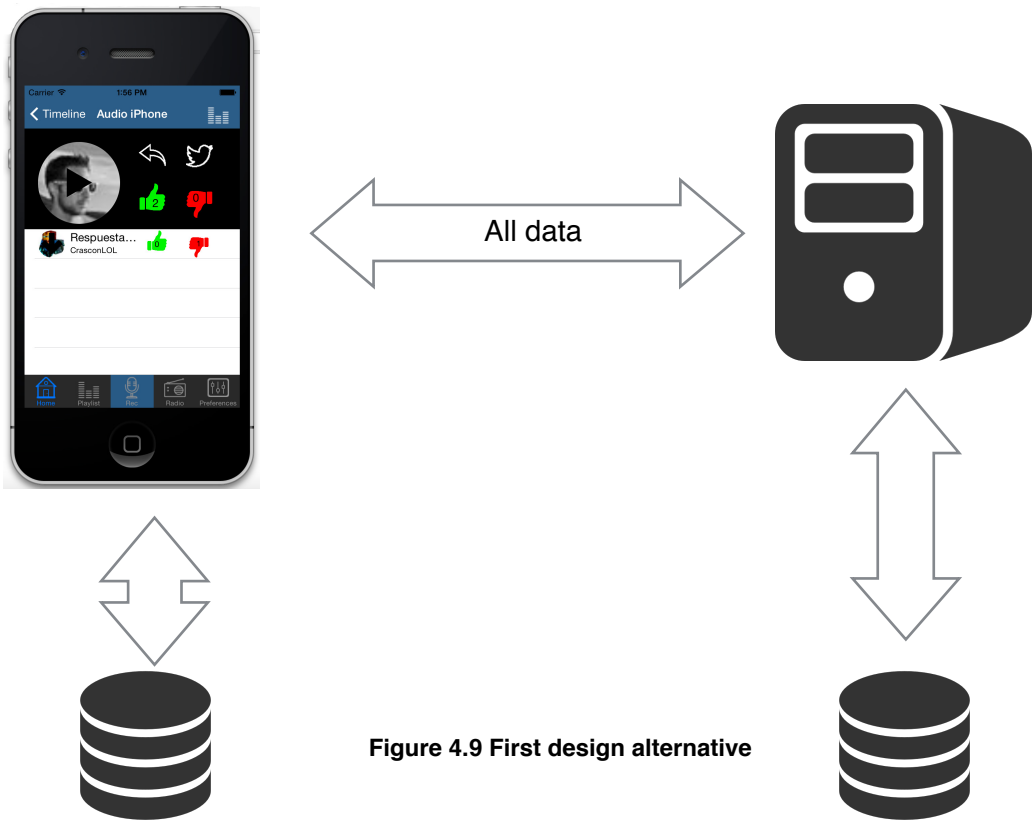


Figure 4.9 First design alternative

Second design alternative

The second alternative take advantage of the web service processing power, the server retrieve only the data that the app will use. This way even with very large amount of data the app performance will be grate and the user experience will be perfect. This is the alternative chosen for this project.

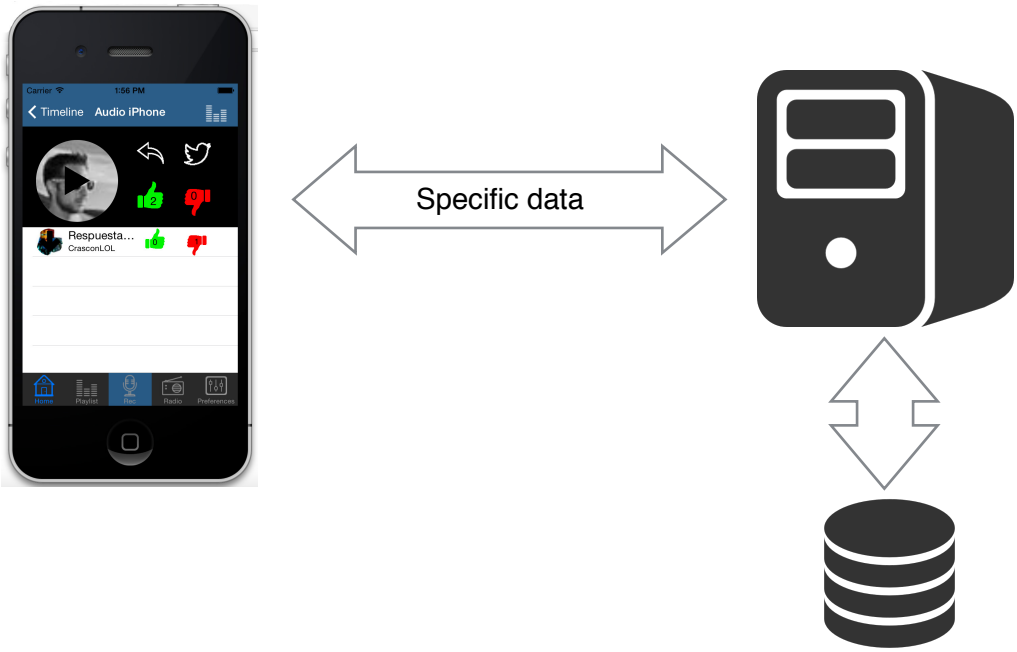


Figure 4.10 Second design alternative

4.1.3 Components and classes diagrams

Following the final design, the app has been modeled using UML diagrams. The diagrams used in this section are components diagram and classes diagram.

Components diagram model the software architecture of the application, showing the application divided in groups and the relations provider-consumer between components.

Controller: This component group the three components that implement the logic of the application. It have the port UI Sender used by user interface to send the actual state to Audio and Options components. The Audio component obtain info from the server using the port Data.

Audio: This component, inside the Controller, include the classes that provide most of the functionalities of the application. This classes controls the reproduction of the audio, record of new audios…

The Audio component communicates with this other components:

- Support, to use frameworks to implement main functionalities.

- Options, to share information about the audios and implement secondary functionalities.

- User Interface, to share information about the state of the interface.

- Server, to obtain data of the audios stored in the server and upload new audios to the server.

Implement Audio Receiver port to allow modifications of the audio information from Options.

Options: This component, inside the Controller, include the classes that provide secondary functionalities of the application. This classes controls votes of the audios, radios…

The Options component communicates with this other components:

- Support, to use frameworks to implement its functionalities.

- Audio, to obtain information of the audios.

- User Interface, to share information about the state of the interface.

Implements Options Receiver port to obtain info from Audio component.

Support: This component, inside the Controller, include frameworks of the operative system that allow the implementation of the functionalities. Implements Utilities port to provide its services to other components.

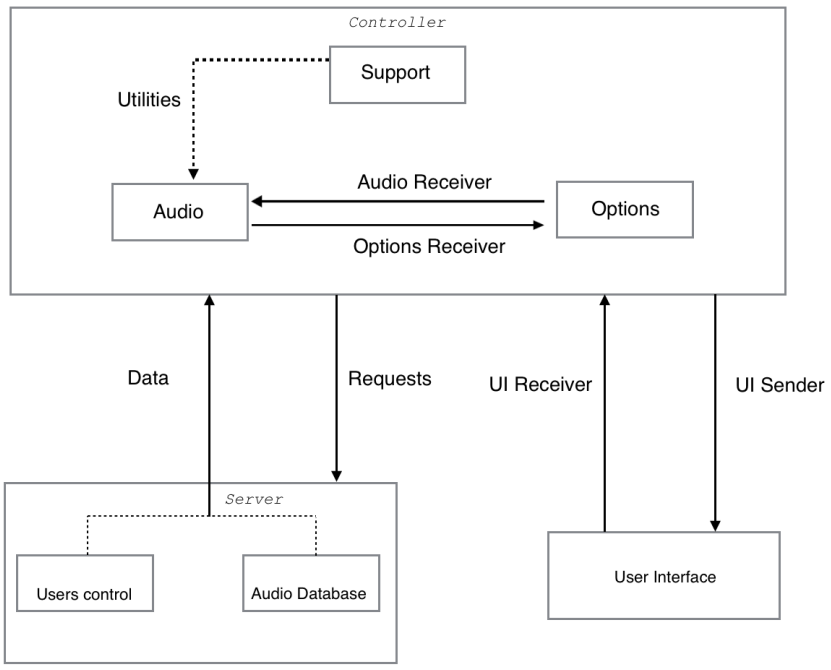


Figure 4.11 Components diagram

User Interface: Includes classes that implement user interfaces. It share information with Audio and Options about state of the interface. Implements UI Receiver port and UI Sender to perform this communication.

Server: This packet group components that include audio data base and user control. Implements Request port to allow requests from the Controller.

From components diagram can be deduced that the application have been design following the standard Model-View-Controller or MVC. This standard followed in most of iOS applications, consist on, a controller that collect user interactions with the views and update the views depending on the actions. In addition, the controller, interacts with the model, which provide information and data, to perform some actions.

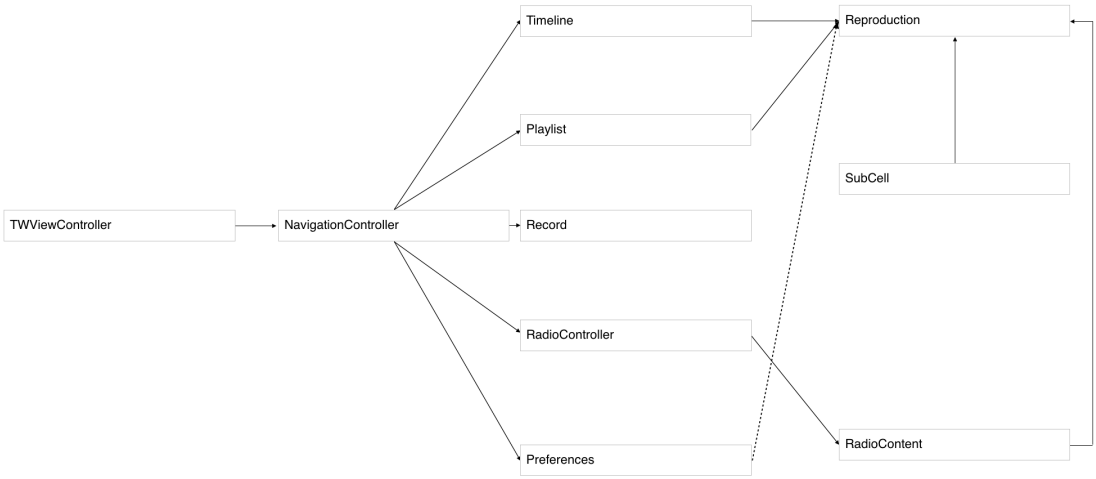


Figure 4.12 Classes diagram

TWViewController
MyManager *myManager UIPickerView *accountPicker NSString *userId
- (void)viewDidLoad -(void) obtainTWInfoOfUser: (NSString *)user

Class TWViewController implements the login of the application. Obtain Twitter accounts of the iPhone and allow the user chose which account wants to use to login in the application. Attributes defined allow the selection of the account as well as share this information to the rest of classes. Methods implemented obtain accounts from the iPhone and obtain information about the account selected.

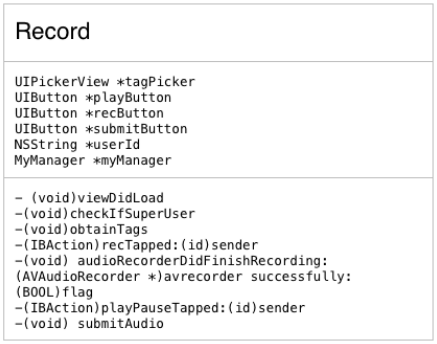
Class *NavigationController* implements the navigation between all the views, allowing going back to previous view and select the different tabs.

NavigationController
- (void)viewDidLoad

Timeline
NSMutableDictionary *userProfilePhotos UITableView *timelineTable NSArray *audioJsons MyManager *myManager
- (void)viewDidLoad -(void)obtainInfo -(void)refreshing -(void) obtainTWInfoOfUser: (NSString *)user -(void)prepareReproduction: (Reproduction *)rep toReproduceAudio: (NSString *)audio withJson: (NSDictionary *)cellJson withIndexPath: (NSIndexPath *) iP

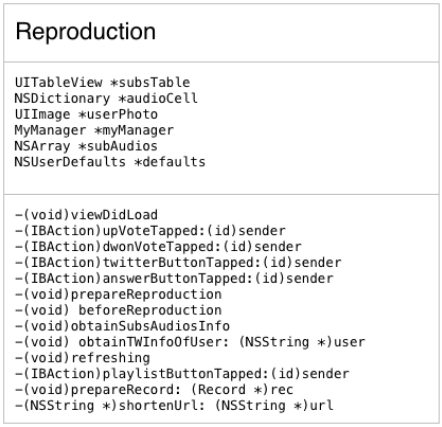
Class *Timeline* implements one of the most important features, this class obtain all data from the server and show the users all available audios, as well as obtain authors info to identify the audios. Attributes defined store user profile photos and audio data. Methods implemented obtain info of authors, obtain audio data form the server and prepare the audio selected to reproduce it in Reproduction view.

Class *Playlist* implements the function with the same name, allow the user order some selected audios to play after. Attributes defined store user profile photo and audio data of selected audios. Methods implemented obtain audio data from selected audios as well as prepare the reproduction of the audios in Reproduction view.



Class *Record* implements one of the most important features, this class allow the user to record new audios or answer to an existing audio, as well as classify the audio and set a title for it. Attributes defined manage the controls of the interface and manage audio data. Methods implemented manage recording and upload process.

Class *Preferences* implements the view where users can set some options about the operation of the app. Attributes defined manage interface controls and allow sharing the preferences with all classes. Methods implemented manage interface interactions connections with other classes to apply preferences.



Class *Reproduction* implements reproduction of the selected audio as well as voting, answer and share process. Attributes defined manage audio data and answer audios data. Methods implemented manage reproduction and interface controls.

4.1.4 Sequences diagrams

In this section sequence diagrams attend developers as a model to implement relations between application objects.

First diagram expose methods executed when users init the app and the login process.

1- Load the interface.

2- Obtain Twitter accounts information from the iPhone.

3- Set accounts on UIPickerView to allow the user select one.

4- After the selection of the account, application connects to Twitter serves to obtain information of the account.

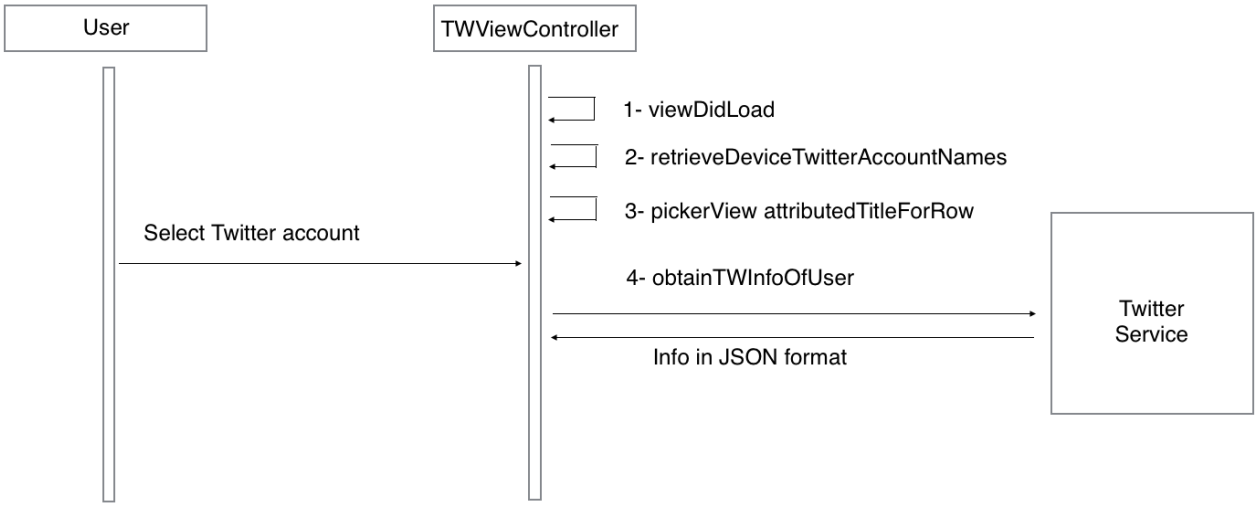


Figure 4.13 First sequence diagram

Second diagram expose the process followed by Timeline class to obtain audio data , show it to the user and the reproduction of an audio.

1- Init the connection to application server and download the information in JSON format.

2- Refresh the interface with the new information.

3- Set information in the TableView.

4- Connect to Twitter server to obtain info from authors and set profile photos in the cells.

User select an audio cell that launch Reproduction view.

5- Set audio info in the view.

6- Init a connection to application server and download answer audios information.

7- Set answer audios information in the TableView.

User press play button.

8- Stop audio player if playing, set new audio data in the audio player and start reproduction.

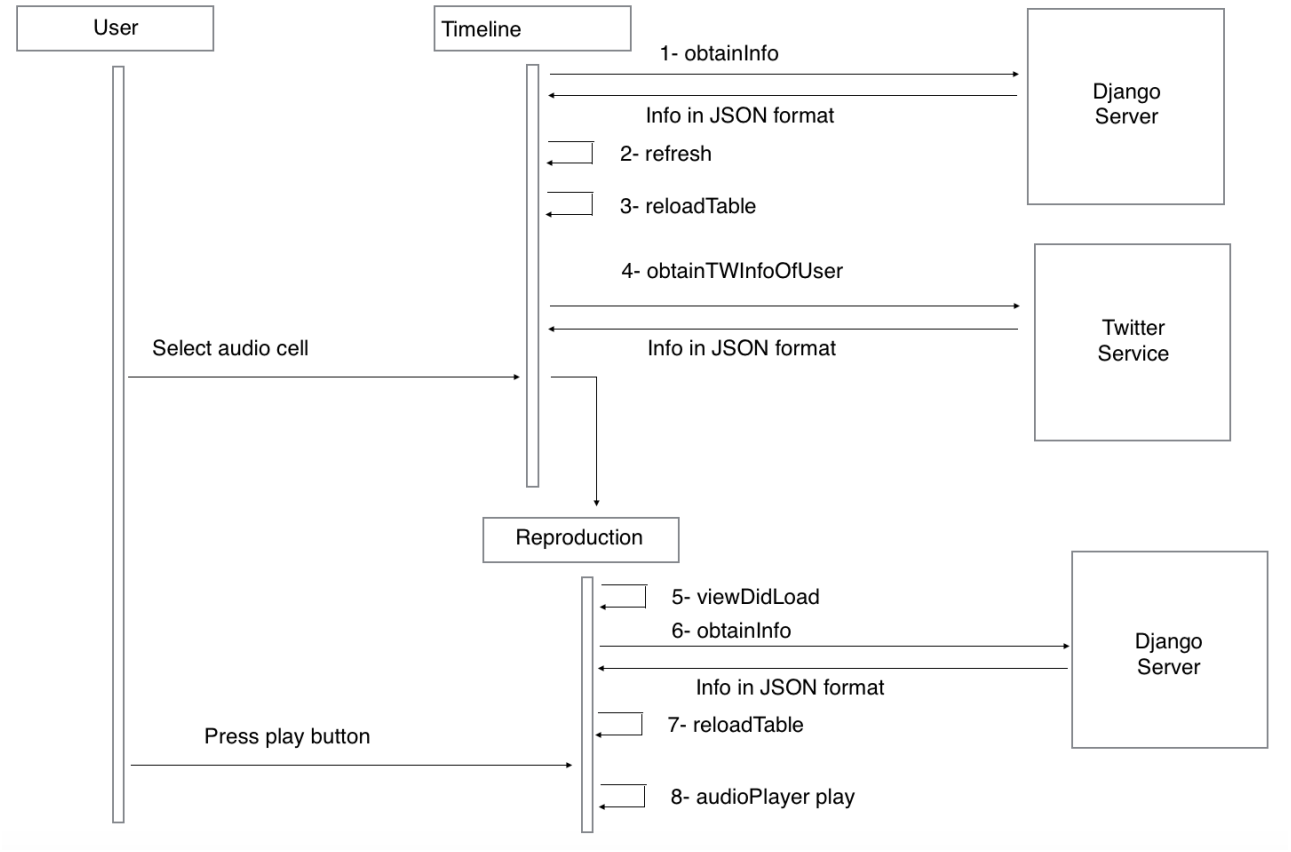


Figure 4.14 Second sequence diagram

Third diagram expose the process followed by Timeline class to obtain audio data , show it to the user and the reproduction of an audio answer.

1- Init the connection to application server and download the information in JSON format.

2- Refresh the interface with the new information.

3- Set information in the TableView.

4- Connect to Twitter server to obtain info from authors and set profile photos in the cells.

User select an audio cell that launch Reproduction view.

5- Set audio info in the view.

6- Init a connection to application server and download answer audios information.

7- Set answer audios information in the TableView.

User press audio answer cell.

8- Connect to application server and download audio answer.

User press play button.

9- Stop audio player if playing, set new audio data in the audio player and start reproduction.

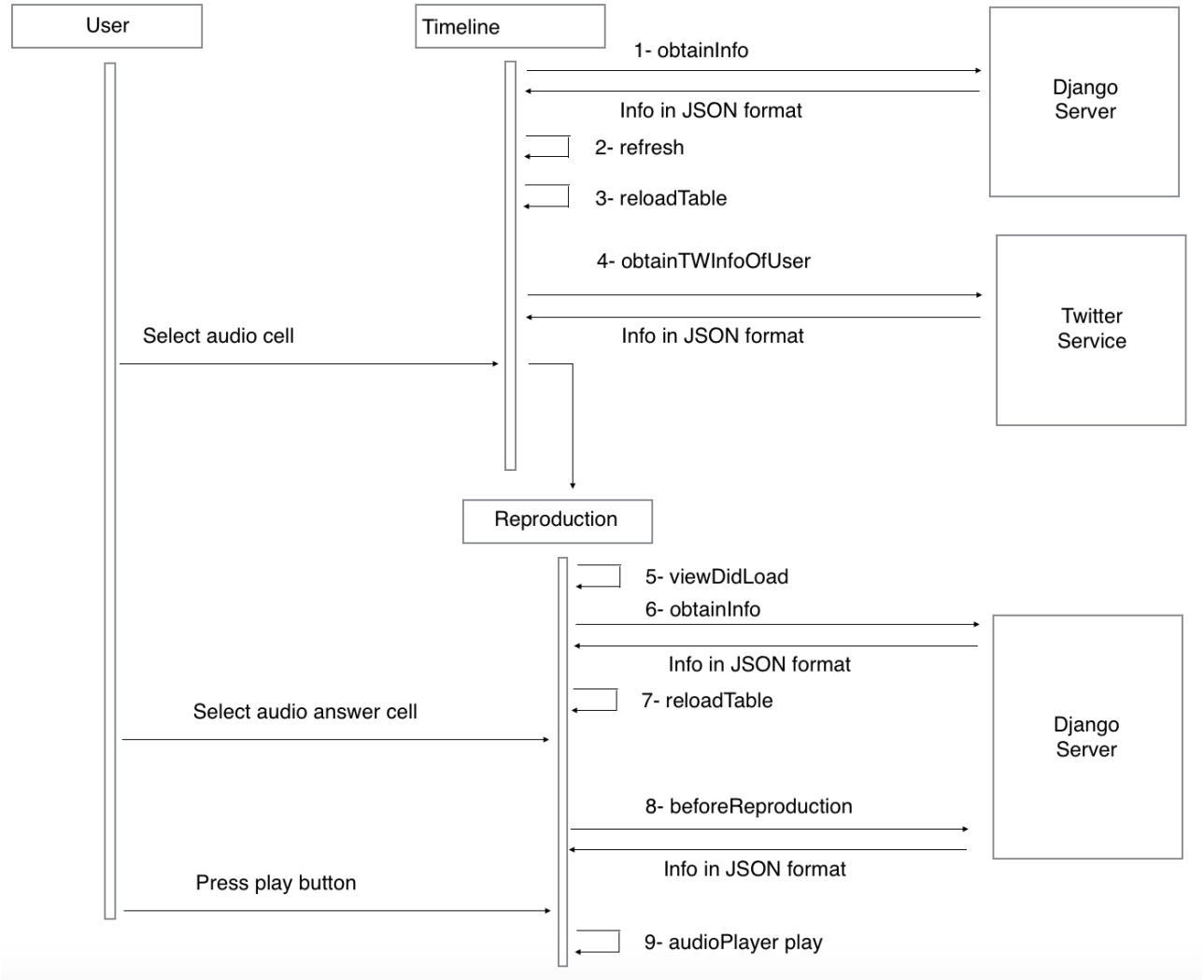


Figure 4.15 Third sequence diagram

Fourth diagram expose the process followed by the application to record an audio or and answer and to upload it.

1- Create the path for the new audio.

2- Setup audio session and setup a new recorder, encoding in AAC, sample rate 44100 and 2 channels (stereo).

User press rec button.

3- Start recording.

User press stop rec button.

4- Stop recording.

5- Set audio player to allow user listen the record before uploading.

6- Enable play and submit buttons.

User select the tag and press submit button.

7- Prepare uploading data and launch alert to ask for the title.

User introduce a title for the audio and press upload.

8- Init the connection with application server, upload audio data and information and check that there is no errors.

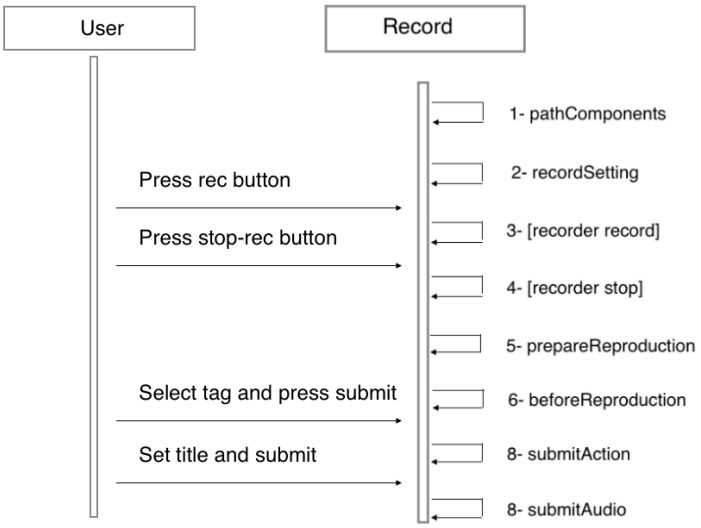


Figure 4.16 Fourth sequence diagram

Fifth diagram expose the process followed by the application to obtain radios information and show it to the user, in Radio view.

1- Init connection to application server and and download radios information.
2- Refresh the interface with the new information.
3- Set information in the TableView.
4- Connect to Twitter server to obtain info from authors and set profile photos in the cells.

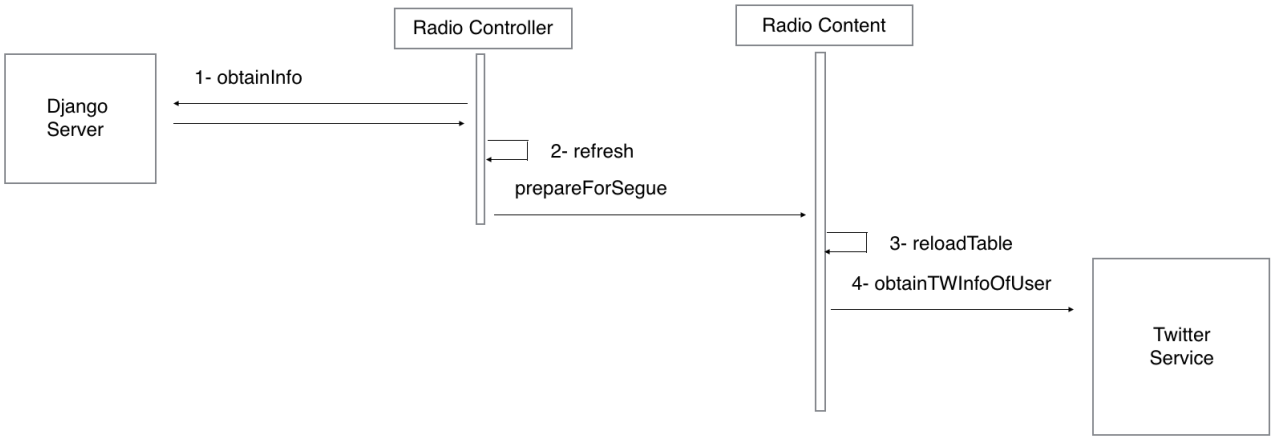


Figure 4.17 Fifth sequence diagram

Last diagram expose the process followed by the application to manage user preferences.

1- Load actual user preferences.
2- Set actual preferences in the view.
User change something in preferences.

3- App store the new preference.

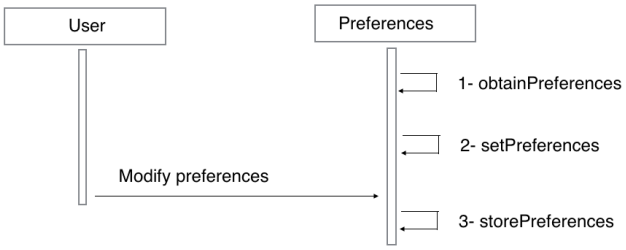


Figure 4.18 Last sequence diagram

4.2 Server design

On this chapter there will be explain the design decisions taken to implement the application web server. As well as the performance of the server and interactions between the server and the application.

First of all the main decision implies chose the framework to implement the backend. The language chosen is Django(Python), actually Django is one of the most used framework to implement backends to mobile applications.

Django is a web development framework based on MVC model. It was developed by World Company, firstly it was designed to manage news webs, and finally was launched as open source in 2005.

Django implements natively CRUD and file storage. To implement the database manage internally creation of necessary structures depending in the configuration, this gives to the developer many possibilities and makes Django highly scalable.

Django is implemented in Python and the philosophy of reusing code. Thanks to the popularity of Django, there are al lot of extensions designed for it. In fact to develop this backend the extension chosen is django-rest-framework. This extension is based on object orientation programing, which is so useful in this project, regarding that the wen service will manage mainly audios.

4.2.1 Database design

Django handle different models of database, for this project the model chosen is SQLite. This model is a system of databases of small size, that store all information (definitions, tables, indexes) in just one portable file. Information is stored in the database and audio file is stored natively with references to them in the database.

Django uses Models to define basic type of data and let the framework create the structures depending on the configuration files. For this project four models have been designed.

MainItem: This model represent main audios of the applications and all information related with them.

```
class MainItem(models.Model):
	contents = models.FileField(upload_to='documents/%Y/%m/%d')
	user = models.CharField(max_length=200)
	userid = models.IntegerField()
	timestamp = models.DateTimeField(auto_now=True)
	upvotes = models.IntegerField(default = 0)
	downvotes = models.IntegerField(default = 0)
	tags = models.ManyToManyField(Tag)
	description = models.CharField(max_length=200)
	def upvote(self):
		self.upvotes = self.upvotes + 1;

	def downvote(self):
		self.downvotes = self.downvotes + 1;

	def upvoteb(self):
		self.upvotes = self.upvotes - 1;

	def downvoteb(self):
		self.downvotes = self.downvotes - 1;
```

SubItem: This model represent audio answers of the applications and all information related with them.

```
class SubItem(models.Model):
	contents = models.FileField(upload_to='documents/%Y/%m/%d')
	user = models.CharField(max_length=200)
	userid = models.IntegerField()
	timestamp = models.DateTimeField(auto_now=True)
	upvotes = models.IntegerField(default = 0)
	downvotes = models.IntegerField(default = 0)
	tags = models.ManyToManyField(Tag)
	description = models.CharField(max_length=200)
	parent = models.ForeignKey('MainItem')

	def upvote(self):
		self.upvotes = self.upvotes + 1;

	def downvote(self):
		self.downvotes = self.downvotes + 1;

	def upvoteb(self):
		self.upvotes = self.upvotes - 1;

	def downvoteb(self):
		self.downvotes = self.downvotes - 1;
```

Tag: This model represent audio tags of the applications and all information related with them.

```
class Tag(models.Model):
```

```
	keyword = models.CharField(max_length=200)
```

Superuser: This model represent users that can upload main audios and all information related with them.

```
class superUser(models.Model):
	user = models.CharField(max_length=200)
	userid = models.IntegerField()
```

4.2.2 Urls and serializers

Following Django philosophy, to access methods there have to be some urls defined. This methods should return information to the application, so this methods have to use Serializers to transform model objects to data understandable by the app, in this project, Serializers transform model object to JSON syntax.

```
url(r'^', include(router.urls)),
url(r'^api-auth/', include('rest_framework.urls', namespace='rest_framework')),

url(r'^mains/(?P<pk>[0-9]+)/upvote$', 'blogvoice.views.main_item_vote'),
url(r'^mains/(?P<pk>[0-9]+)/downvote$', 'blogvoice.views.main_item_downvote'),

url(r'^subs/(?P<pk>[0-9]+)/upvote$', 'blogvoice.views.sub_item_vote'),
url(r'^subs/(?P<pk>[0-9]+)/downvote$', 'blogvoice.views.sub_item_downvote'),

url(r'^mains/(?P<pk>[0-9]+)/upvoteb$', 'blogvoice.views.main_item_vote_b'),
url(r'^mains/(?P<pk>[0-9]+)/downvoteb$', 'blogvoice.views.main_item_downvote_b'),

url(r'^subs/(?P<pk>[0-9]+)/upvoteb$', 'blogvoice.views.sub_item_vote_b'),
url(r'^subs/(?P<pk>[0-9]+)/downvoteb$', 'blogvoice.views.sub_item_downvote_b'),

url(r'^searchSubsOfMain/(?P<pk>[0-9]+)$', 'blogvoice.views.searchSubsOfMain'),
url(r'^first10Tag/(?P<pk>[0-9]+)$', 'blogvoice.views.first10Tag'),
url(r'^searchUser/(?P<pk>[0-9]+)$', 'blogvoice.views.searchUser'),
url(r'^first10Subs/(?P<pk>[0-9]+)$', `blogvoice.views.first10Subs'),
```

In section 5.3 and 5.4 there will be explanations about the different methods, and how the application use them.

```
class HyperlinkedFileField(serializers.FileField):
	def to_native(self, value):
		request = self.context.get('request', None)
		protocol = 'https://' if request.is_secure() else 'http://'
		return ''.join([protocol, request.get_host(), '/bvs/', value.url])

class MainItemSerializer(serializers.HyperlinkedModelSerializer):
	contents = HyperlinkedFileField()

	class Meta:
		model = MainItem
		fields = ('id', 'description', 'user', 'userid', 'timestamp', 'upvotes',
'downvotes', 'tags', 'url', 'contents')
		#depth = 1

class TagSerializer(serializers.HyperlinkedModelSerializer):
```

```
        class Meta:
                model = Tag
                fields = ('id', 'keyword', 'url')
```

```
class SubItemSerializer(serializers.HyperlinkedModelSerializer):
        contents = HyperlinkedFileField()
        parent_id = serializers.Field('parent.id')

        class Meta:
                model = SubItem
                fields = ('id', 'parent', 'parent_id', 'description', 'user', 'userid',
'timestamp', 'upvotes', 'downvotes', 'tags', 'url', 'contents')

class superUserSerializer(serializers.HyperlinkedModelSerializer):
        class Meta:
                model = superUser
                fields = ('user', 'userid')

class MainItemSerializer2(serializers.ModelSerializer):
        class Meta:
                model = MainItem
                fields = ('id', 'description', 'user', 'userid', 'timestamp', 'upvotes',
'downvotes', 'tags', 'contents')

class SubItemSerializer2(serializers.ModelSerializer):
        parent_id = serializers.Field('parent.id')
        class Meta:
                model = SubItem
                fields = ('id', 'parent', 'parent_id', 'description', 'user', 'userid',
'timestamp', 'upvotes', 'downvotes', 'tags', `contents')
```

4.2.3 GET methods

SearchSubsOfMain: This method return in JSON format all answer audios of a main audio. Main audio is identify by the parameter *pk* which is the main audio id. All the answer audios are ordered by date.

```
@api_view(['GET'])
def searchSubsOfMain(request, pk):
        lista = SubItem.objects.filter(parent_id=pk)
        serializer = SubItemSerializer2(lista)
        content = JSONRenderer().render(serializer.data)
        return HttpResponse(json.dumps(serializer.data), content_type="application/json")
```

First10Subs: This method return in JSON format ten answer audios of a main audio. Main audio is identify by the parameter *pk* which is the main audio id. The answer audios are ordered by positive votes, being the ten most voted answers.

```
@api_view(['GET'])
def first10Subs(request, pk):
        lista = SubItem.objects.filter(parent_id=pk).order_by('upvotes').reverse()
        serializer = SubItemSerializer2(lista)
```

```
        content = JSONRenderer().render(serializer.data)
        return HttpResponse(json.dumps(serializer.data), content_type="application/json")
```

First10Tag: This method return in JSON format ten main audios of a specific tag. Tag is identify by the parameter *pk* which is the tag id. The answer audios are ordered by positive votes, being the ten most voted audios of the specific tag.

```
@api_view(['GET'])
def first10Tag(request, pk):
        lista = MainItem.objects.filter(tags=pk).order_by('upvotes').reverse()
        serializer = MainItemSerializer2(lista)
        content = JSONRenderer().render(serializer.data)
        return HttpResponse(json.dumps(serializer.data), content_type="application/json")
```

SearchUser: This method return in JSON format if the user is in the list of super-users or not. User is identify by the parameter *pk* which is the Twitter user id. The answer is True or False.

```
@api_view(['GET'])
def searchUser(request, pk):
        list = superUser.objects.filter(userid=pk)
        if not list:
                response_data = {}
                response_data['result'] = 'False'
                return  HttpResponse(json.dumps(response_data),  content_type="application/
json")
        response_data = {}
        response_data['result'] = 'True'
        return HttpResponse(json.dumps(response_data), content_type="application/json")
```

4.2.4 POST methods

Vote: This methods add one positive vote to a main or a sub audio. Main or sub audios is identify by parameter *pk* which id the id.

```
@api_view(['GET'])
def main_item_vote(request, pk, format=None):
        try:
                main = MainItem.objects.get(pk=pk)
        except:
                return Response(status=status.HTTP_404_NOT_FOUND)
        main.upvote()
        main.save()
        return Response(status=status.HTTP_204_NO_CONTENT)

@api_view(['GET'])
def sub_item_vote(request, pk, format=None):
        try:
                sub = SubItem.objects.get(pk=pk)
        except:
                return Response(status=status.HTTP_404_NOT_FOUND)
        sub.upvote()
```

```
        sub.save()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

DownVote: This methods add one negative vote to a main or a sub audio. Main or sub audios is identify by parameter *pk* which id the id.

```
@api_view(['GET'])
def main_item_downvote(request, pk, format=None):
        try:
                main = MainItem.objects.get(pk=pk)
        except:
                return Response(status=status.HTTP_404_NOT_FOUND)
        main.downvote()
        main.save()
        return Response(status=status.HTTP_204_NO_CONTENT)

@api_view(['GET'])
def sub_item_downvote(request, pk, format=None):
        try:
                sub = SubItem.objects.get(pk=pk)
        except:
                return Response(status=status.HTTP_404_NOT_FOUND)
        sub.downvote()
        sub.save()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

Un-Vote: This methods remove one positive vote to a main or a sub audio. Main or sub audios is identify by parameter *pk* which id the id.

```
@api_view(['GET'])
def main_item_vote_b(request, pk, format=None):
        try:
                main = MainItem.objects.get(pk=pk)
        except:
                return Response(status=status.HTTP_404_NOT_FOUND)
        main.upvoteb()
        main.save()
        return Response(status=status.HTTP_204_NO_CONTENT)

@api_view(['GET'])
def sub_item_vote_b(request, pk, format=None):
        try:
                sub = SubItem.objects.get(pk=pk)
        except:
                return Response(status=status.HTTP_404_NOT_FOUND)
        sub.upvoteb()
        sub.save()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

Un-DownVote: This methods remove one negative vote to a main or a sub audio. Main or sub audios is identify by parameter *pk* which id the id.

```
@api_view(['GET'])
def main_item_downvote_b(request, pk, format=None):
	try:
		main = MainItem.objects.get(pk=pk)
	except:
		return Response(status=status.HTTP_404_NOT_FOUND)
	main.downvoteb()
	main.save()
	return Response(status=status.HTTP_204_NO_CONTENT)

@api_view(['GET'])
def sub_item_downvote_b(request, pk, format=None):
	try:
		sub = SubItem.objects.get(pk=pk)
	except:
		return Response(status=status.HTTP_404_NOT_FOUND)
	sub.downvoteb()
	sub.save()
	return Response(status=status.HTTP_204_NO_CONTENT)
```

Chapter 5: Application testing

Chapter 5: Application testing

In this chapter there are some test performed to the application, to verify that all functionalities expected to work, do it correctly. This test corresponds with actions that a user will do using the application.

5.1 Conditions before test

In order to perform good test to the app, the test environment have to fulfill some conditions:

- Free Voices server have to contain some audios in order to allow audio tests.
- iPhone have to have set at least one Twitter account.
- To perform tests related with super-user, iPhone have to have set at least one Twitter account of a super-user.
- iPhone have to have a stable connection to Internet.

5.2 Performed tests

Before performing the tests the server and the iPhone used (iPhone 4S, iOS 7.1) were configured with the following presets:

- The server contain 2 main audios, one uploaded from an iPhone called “Audio iPhone”, the other one uploaded from Android app called “Audio Android”.
- The server contain 2 answer audios, one per main audio, called “Answer iPhone” and “Answer Android”.
- The iPhone is connected to a stable Wi-Fi.
- The iPhone have 2 Twitter accounts one of them of a user registered in Free Voices server as super-user.

In order to follow the methodology SCRUM, each test is described in a table with the following format:

ID	
NAME	
OBJECTIVE	
STEPS SECUENCE	
POSIBLE ERRORS	
RESULT	
STATE	

Table 5.1 Table format for applications tests

Each field will have this content:

- ID: User story identifier. With format System Test ST-XXX
- NAME: Name of the test. The name summarize the test, the action, the complements, and who perform the action.
- OBJECTIVE: Description of the test to be performed and the functionality that will be test.
- STEPS SEQUENCE: Steps that the user has to perform to reach the functionality.
- POSSIBLE ERRORS: Errors that can appear during the test.
- RESULT: Result of the test after following the steps proposed.
- STATE: Pass or Fail, indicate if the test was right or not.

ID	**ST-001**
NAME	Super-user login in the app and listen to any audio
OBJECTIVE	A user registered as super-user in the server try to login in the application and listen to any audio.
STEPS SECUENCE	1- Init the app 2- Select super-usr Twitter account from picker 3- Select “Audio iPhone” in Timeline view 4- Press play button in Reproduction view
POSIBLE ERRORS	- No login - Audios do not appear in Timeline view - Audio do not reproduce after pressing play button
RESULT	The user login correctly, audios appear in Timeline view, audio cell selected launch Reproduction view and the audio is reproduced after pressing the play button.
STATE	PASS

Table 5.2 ST-001 Super-user login

ID	**ST-002**
NAME	Normal user login in the app and listen to any audio
OBJECTIVE	A user non-registered as super-user in the server try to login in the application and listen to any audio.
STEPS SECUENCE	1- Init the app 2- Select Twitter account from picker 3- Select “Audio Android” Timeline view 4- Press play button in Reproduction view
POSIBLE ERRORS	- No login - Audios do not appear in Timeline view - Audio do not reproduce after pressing play button
RESULT	The user login correctly, audios appear in Timeline view, audio cell selected launch Reproduction view and the audio is reproduced after pressing the play button.
STATE	PASS

Table 5.3 ST-002 Normal user login

ID	ST-003
NAME	User login in the app with out Internet connection.
OBJECTIVE	A user try to login in the application with out internet connection, the app should not allow this because it can not identify the user.
STEPS SECUENCE	1- Init the app 2- Select Twitter account from picker
POSIBLE ERRORS	- The app allow the user use the app
RESULT	The app try to connect to Twitter server to identify the user and after a period of not connection, notify the user that needs internet connection to use the app.
STATE	PASS

Table 5.4 ST-003 Login with out Internet connection

ID	ST-004
NAME	Super-user upload a main audio
OBJECTIVE	A user registered as super-user in the server try to upload a main audio.
STEPS SECUENCE	1- Init the app 2- Select super-user Twitter account from picker 3- Select Rec tab 4- Select a tag or the audio from the picker 5- Press rec button to record an audio 6- Press stop rec button to stop recording 7- Press submit button 8- Set a title and upload the audio
POSIBLE ERRORS	- No login - Interface of Rec tab blocked
RESULT	The user login correctly. When the Rec tab is selected the interface is valuable for the user. The user record the audio and upload it.
STATE	PASS

Table 5.5 ST-004 Super-user upload main audio

ID	ST-005
NAME	Super-user upload an audio answer.
OBJECTIVE	A user registered as super-user in the server try to upload an audio answer.

ID	ST-005
STEPS SECUENCE	1- Init the app 2- Select Twitter account from picker 3- Select “Audio iPhone” Timeline view 4- Press answer button 5- Rec tab launched 6- Press rec button to record an audio 7- Press stop rec button to stop recording 8- Press submit button 9- Set a title and upload the audio
POSIBLE ERRORS	- Rec tab not launched - Rec tab interface blocked
RESULT	Rec tab launched and the user follows the recording and uploading process with out any problem.
STATE	PASS

Table 5.6 ST-005 Super-user upload audio answer

ID	ST-006
NAME	Normal user upload an audio answer.
OBJECTIVE	A user non-registered as super-user in the server try to upload an audio answer.
STEPS SECUENCE	1- Init the app 2- Select Twitter account from picker 3- Select “Audio Android” in Timeline view 4- Press answer button 5- Rec tab launched 6- Press rec button to record an audio 7- Press stop rec button to stop recording 8- Press submit button 9- Set a title and upload the audio
POSIBLE ERRORS	- Rec tab not launched - Rec tab interface blocked
RESULT	Rec tab launched and the user follows the recording and uploading process with out any problem.
STATE	PASS

Table 5.7 ST-006 Normal user upload audio answer

ID	ST-007
NAME	User try to listen to an audio answer.
OBJECTIVE	A user try to upload an audio answer of an specific main audio.
STEPS SECUENCE	1- Init the app 2- Select Twitter account from picker 3- Select “Audio iPhone” in Timeline view 4- Press “Answer iPhone” cell to download the audio 5- Press play button

ID	ST-007
POSIBLE ERRORS	- No download button in the answer audio - No audio downloaded after pressing the button - No reproduction of the audio
RESULT	The audio answer was downloaded and reproduced with out any problem
STATE	PASS

Table 5.8 ST-007 User listen to an audio answer

ID	ST-008
NAME	User reproduce an audio from a radio.
OBJECTIVE	A user try to reproduce an audio from a radio promoted by the app.
STEPS SECUENCE	1- Init the app 2- Select Twitter account from picker 3- Select Radio tab 4- Select “Audio Android” in Radio view to launch Reproduction view 5- Press play button
POSIBLE ERRORS	- Reproduction view not launched
RESULT	Audio cell from Radio view launch Reproduction view and produce the audio correctly.
STATE	PASS

Table 5.9 ST-008 User use radio

ID	ST-009
NAME	User change between radios.
OBJECTIVE	User try to change between different radios in Radio view.
STEPS SECUENCE	1- Init the app 2- Select Twitter account from picker 3- Select Radio tab 4- Swipe right or left to change between radios
POSIBLE ERRORS	- App do not reacts to swipe gesture - The tactile of the radios is wrong
RESULT	App reacts to swipe gesture and change title and content when change the radio
STATE	PASS

Table 5.10 ST-009 User change between radios

ID	ST-010
NAME	User select all radios in preferences.
OBJECTIVE	User try to select all available radios in Preferences view.
STEPS SECUENCE	1- Init the app 2- Select Twitter account from picker 3- Select Preferences view 4- Select all radios 5- App notify the user to reset the app in order to apply the new preferences 6- Reset the app 7- Select Radio tab
POSIBLE ERRORS	- App not notify the user to reset the app - App can not init after the reset - Do not appear all radios in Radio view
RESULT	App notify the user to reset the app, app init again after the reset and all radios are available in Radio view.
STATE	PASS

Table 5.11 ST-010 User select all radios

ID	ST-011
NAME	User select no radios in preferences.
OBJECTIVE	User try to select no radios in Preferences view.
STEPS SECUENCE	1- Init the app 2- Select Twitter account and login 3- Select Preferences view 4- Select all radios 5- App notify the user to reset the app in order to apply the new preferences 6- Reset the app 7- Select Radio tab
POSIBLE ERRORS	- App not notify the user to reset the app - App can not init after the reset - App crash when Radio tab is selected
RESULT	App notify the user to reset the app, app init again after the reset and Radio view is empty.
STATE	PASS

Table 5.12 ST-011 User select no radios

Chapter 6: Conclusions and future improvements

Chapter 6: Conclusions and future lines

This chapter collect all the conclusions obtained during the development of the project and the author experiences, as well as the future improvements with new functionalities that improve the application.

6.1 Conclusions

Nowadays, society is completely immersed in a technologic world, people need to know everything about the world wrapping them. Technology helps a lot covering this demand of information, but this demand has some many ways to be fulfill, and technology evolve everyday to satisfy users claims. This project is looking forward to provide a new innovative way of share information and at the same time a reaction from users to interact with the platform in a way that no other media allow. The possibility of creating spoken conversations between users and authors is completely new and bring close users with authors and vice versa.

This final degree project have been developed at the same time with a similar project but form Android platform, this adds to the project more range and allow interaction between users and authors from both platforms. This project have been promoted by the group Software Engineering Lab from Informatics Department and Journalist and Audiovisual Communication Department . The cooperation of people involve in the project have been fundamental to the development and launching of this first approach to the idea.

During the development the application could take the direction of social network, but this was not the purpose of the platform. This project is searching for the interaction between users but not in the same way as in a social network. As well, during the development, involve people have realize that the platform with some little changes can be redirect to be a platform where new musicians can share their songs and some other similar variants.

The experience of the author,during the development of the project, have been very positive. The possibility of performing the project in a real scenario and the collaboration with other people from SEL has been so helpful as a preparation for a future job. Mobile technologic have been increasing since some years and will continue growing, actually changed how society behave and this will continue advancing and changing lives.

6.2 Future Lines

In this section, future lines are proposed to improve some functionalities and add ones more.

6.2.1 Student Test

There is a test programed do to with some student from a subject of radio, that couldn't be performed yet because the subject finished before the app was fully ready.

This test is looking for test the app in a real situation, some students will be super-users and will create content in the app, and some other students will be normal users and will proceed as listeners commenting the audios.

6.2.2 Offline reproduction

The improvement of offline reproduction of audios is thought to allow users with no continuous connection to Internet, download the audios that want to listen to and listen to them when they do not have connection.

To perform this improvement there will be a data base of downloaded audios inside the iPhone that allow users add and remove audios.

6.2.3 Radio streaming

This improve consist on implement in the web service a streaming with the most voted audios and establish some hour in which the streaming works. This functionality is very similar to actual functionality of Radio but the streaming adds some features, like chose exactly in which hours users list to the audios.

This improve could provide a business model, between audios can be added some advertisements and promotions.

6.2.4 Application for Windows Phone

Actually Windows Phone is the third most used platform and since the app have been developed for Android and iOS platform, and important improve to contribute in the resolution of the problem exposed in this project, is the implementation of the Windows Phone app.

Bibliography

[1] mobiThinking.com “Global mobile statistics 2013 Section E: Mobile apps, app stores, pricing and failure rates” [Online Document] May 2013.
http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/e#appdownloads

[2] Wikipedia “JSON” [Online Document] 12/6/2014
http://en.wikipedia.org/wiki/JSON

[3] Wikipedia “MP3” [Online Document] 6/6/2014
http://es.wikipedia.org/wiki/MP3

[4] CuaQea Development team “cuaQea” [Android app] 4/4/2014
https://play.google.com/store/apps/details?id=com.cuaqea.cuaqea&hl=us

[5] PMobile “Blaving” [Android app] 19/10/2012
https://play.google.com/store/apps/details?id=com.deciloo.application

[6] Soundcloud ltd “Soundcloud” [iOS app] 6/5/2014
https://itunes.apple.com/es/app/soundcloud/id336353151?mt=8

[7] TuneIn “TuneIn” [iOS app] 29/5/2014
https://itunes.apple.com/es/app/tunein-radio/id418987775?mt=8

[8] Audiovoo ltd “Audiovoo” [iOS app] 24/4/2014
https://itunes.apple.com/es/app/audioboo/id305204540?mt=8

[9] Union Radio - Prisa “CADENA SER for iPhone” [iOS app] 25/1/2014
https://itunes.apple.com/us/app/cadena-ser-para-iphone/id401987596?mt=8

[10] Radio Popular S.A - COPE “COPE Radio for iPhone” [iOS app] 17/5/2014
https://itunes.apple.com/us/app/radio-cope/id591046636?mt=8

[11] Wikipedia “History of iOS” [Online Document] 17/6/2014
http://en.wikipedia.org/wiki/Ios_version_history

[12] Ray, J. “iPhone Application Development in 24 hours” Second Edition. [Online Document], 15/10/2011.
http://proquest.safaribooksonline.com/book/programming/mobile/9780132601214

[13] Sadun, E. “The iOS 5 Developer’s Cookbook: Core Concepts and Essential Recipes for iOS Programmers”, Third Edition. Addison-Wesley Professional. 14/11/2011. [Online Document],
http://proquest.safaribooksonline.com/book/programming/mobile/9780132613477

[14] Apple Developer. “Developer Tools”. [Online Document].
https://developer.apple.com/technologies/tools/

[15] Apple Developer. “The Objective-C Programming Language”. [Online Document] 12/10/2011.
https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/Objec tiveC/Introduction/introObjectiveC.html

[16] SOFTENG. “Metodología Scrum”. [Online Document].
http://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologia- scrum.html